NNT/NL : 2020AIXM0001/001ED000

THÈSE DE DOCTORAT

Soutenue à Aix-Marseille Université
le 3 Octobre 2023 par

Lauri LAATU

Development of artificial intelligence algorithms adapted to big data processing in embedded (FPGAs) trigger and data acquisition systems at the LHC

Discipline
Physique et Sciences de la Matière

Spécialité
Physiques des Particules et Astroparticules

École doctorale
ED 352 Physique et Sciences de la Matière

Laboratoire/Partenaires de recherche
Centre de Physique des Particules de Marseille (CPPM) Aix-Marseille Université

Composition du jury

Julie MALCLES CEA Saclay IRFU, CEA, Université Paris-Saclay	Rapporteure
Julien DONINI LPC, Université Clermont Auvergne, CNRS/IN2P3	Rapporteur
Christophe OCHANDO LLR, Institut Polytechnique de Paris	Examinateur
Cristinel DIACONU CPPM, Aix Marseille Université, CNRS/IN2P3	Président du jury
Emmanuel MONNIER CPPM, Aix Marseille Université, CNRS/IN2P3	Directeur de thèse
Georges AAD CPPM, Aix Marseille Université, CNRS/IN2P3	Co-directeur de thèse

Affidavit

Je soussigné, Lauri Laatu, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique de Emmanuel Monnier et Georges Aad, dans le respect des principes d'honnêteté, d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect à la fois de la charte nationale de déontologie des métiers de la recherche et de la charte d'Aix-Marseille Université relative à la lutte contre le plagiat.

Ce travail n'a pas été précédemment soumis en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Fait à Marseille le 26.6.2023

L. Laatu

Liste de publications et participation aux conférences

Liste des publications réalisées dans le cadre du projet de thèse:

1. Georges Aad, Anne-Sophie Berthold, Thomas Calvet, et al. “Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters”. In: Computing and Software for Big Science 5.1 (Oct. 2021), p. 19. ISSN: 2510-2044. DOI: 10.1007/s41781-021-00066-y. URL: https://doi.org/10.1007/s41781-021-00066-y

2. G. Aad, T. Calvet, N. Chiedde, et al. “Firmware implementation of a recurrent neural network for the computation of the energy deposited in the liquid argon calorimeter of the ATLAS experiment”. In: Journal of Instrumentation 18.05 (May 2023)

3. L. Laatu "Recurrent Neural Networks for Real-Time Processing of ATLAS Liquid Argon Calorimeter Signals with FPGAs" in JRJC 2021. Book of Proceedings. Sept. 2022

4. L. Laatu "Machine Learning for Real-Time Processing of ATLAS Liquid Argon Calorimeter Signals with FPGAs" in PoS(EPS-HEP2021)752

Participation aux conférences et écoles d’été au cours de la période de thèse:

1. EPS-HEP 2021

2. Journées de Rencontre des Jeunes Chercheurs 2021

3. Astroinfo 2021

4. School of Statistics 2022

5. HEP-2023

Résumé

Le modèle standard de la physique des particules est achevé après la découverte du boson de Higgs au Grand collisionneur de hadrons (LHC) en 2012. La découverte d'une nouvelle physique au-delà du modèle standard et sonder le secteur du boson de Higgs sont deux des objectifs les plus importants des expériences actuelles et futures de physique des particules. En 2026-2029, le LHC fera l'objet d'une mise à niveau afin d'augmenter sa luminosité instantanée d'un facteur de 5 à 7 par rapport à sa luminosité nominale. Cette mise à niveau marquera le début de l'ère du LHC à haute luminosité (HL-LHC). Parallèlement, les détecteurs ATLAS et CMS seront mis à niveau pour faire face à l'augmentation de la luminosité du LHC.

Le calorimètre à argon liquide (LAr) d'ATLAS mesure l'énergie des particules produites dans les collisions proton-proton au LHC. L'électronique de lecture du calorimètre LAr sera remplacée pour le préparer à la l'ère HL-LHC. Cela lui permettra de fonctionner à un taux de déclenchement plus élevé et d'avoir une granularité accrue au niveau du déclenchement. L'énergie déposée dans le calorimètre LAr est reconstruite à partir du signal d'impulsion électronique à l'aide d'un algorithme de filtrage optimal. L'énergie est calculée en temps réel à l'aide de cartes électroniques basées sur des circuits logique programmables (FPGA). Les FPGA ont été choisis en raison de leur capacité à traiter de grandes quantités de données avec une faible latence, ce qui est une exigence du système de déclenchement d'ATLAS. L'augmentation de la luminosité du LHC entraînera un taux élevé de collisions multiples et simultanées entre protons et protons (pileup), ce qui se traduira par une dégradation significative de la résolution en énergie calculée par les algorithmes de filtrage optimal. Il est extrêmement important de calculer l'énergie avec une grande précision afin d'atteindre les objectifs de physique de l'expérience ATLAS au HL-LHC.

Les récentes avancées dans le domaine de l'apprentissage profond, associées à la capacité de calcul accrue des FPGA, font des algorithmes d'apprentissage profond des outils prometteurs pour remplacer les algorithmes de filtrage optimal existants. Dans cette thèse, des réseaux neuronaux récurrents (RNN) sont développés pour calculer l'énergie déposée dans le calorimètre LAr. Des réseaux récurrents à mémoire court et long terme (LSTM) et des réseaux neuronaux récurrents simples sont étudiés. Les paramètres de ces réseaux sont étudiés en détail afin d'optimiser les performances. Les réseaux développés se révèlent plus performants que les algorithmes de filtrage optimaux. Ces réseaux sont en outre optimisés pour être déployés sur des FPGA par des méthodes de quantification et de compression qui permettent de réduire l'utilisation de ressources avec un effet minimal sur les performances.

Le calorimètre LAr est composé de 182000 canaux individuels pour lesquels nous devons calculer les énergies déposées. L'entraînement de 182000 réseaux neuronaux différents n'est pas réalisable en pratique. Une nouvelle méthode basée sur l'apprentissage non supervisé est développée pour former des groupes de canaux avec des signaux d'impulsion électroniques similaires, ce qui permet d'utiliser le même réseau neuronal pour tous les canaux dans un même groupe. Cette méthode réduit le nombre de réseaux neuronaux nécessaires à environ 100, ce qui permet de couvrir l'ensemble du détecteur avec ces algorithmes avancés.

Mots clés: L'intelligence artificielle, calorimètre à argon liquide, ATLAS, LHC, FPGA

Abstract

The Standard Model of particle physics is completed after the discovery of the Higgs boson at the Large Hadron Collider (LHC) in 2012. Discovering new physics beyond the Standard Model and probing the newly discovered Higgs sector are two of the most important goals of current and future particle physics experiments. In 2026-2029, the LHC will undergo an upgrade to increase its instantaneous luminosity by a factor of 5-7 with respect to its design luminosity. This upgrade will mark the beginning of the High Luminosity LHC (HL-LHC) era. Concurrently, the ATLAS and the CMS detectors will be upgraded to cope with the increased LHC luminosity.

The ATLAS liquid argon (LAr) calorimeter measures the energies of particles produced in proton-proton collisions at the LHC. The LAr calorimeter readout electronics will be replaced to prepare it for the HL-LHC era. This will allow it to run at a higher trigger rate and have increased granularity at the trigger level. The energy deposited in the LAr calorimeter is reconstructed out of the electronic pulse signal using the optimal filtering algorithm. The energy is computed in real-time using custom electronic boards based on Field Programmable Gate Arrays (FPGAs). FPGAs are chosen due to their ability to process large amounts of data with low latency which is a requirement of the ATLAS trigger system. The increased LHC luminosity will lead to a high rate of simultaneous multiple proton-proton collisions (pileup) that results in a significant degradation of the energy resolution computed by the optimal filtering algorithm. Computing the energy with high precision is of utmost importance to achieve the physics goals of the ATLAS experiment at the HL-LHC.

Recent advances in deep learning coupled with the increased computing capacity of FPGAs makes deep learning algorithms promising tools to replace the existing optimal filtering algorithms. In this dissertation, recurrent neural networks (RNNs) are developed to compute the energy deposited in the LAr calorimeter. Long-Short-Term-Memory (LSTM) and simple RNNs are investigated. The parameters of these neural networks are studied in detail to optimize the performance. The developed networks are shown to outperform the optimal filtering algorithms. The models are further optimized to be deployed on FPGAs by quantization and compression methods which are shown to reduce the resource consumption with minimal effect on the performance.

The LAr calorimeter is composed of 182000 individual channels for which we need to compute the deposited energies. Training 182000 different neural networks is not practically feasible. A new method based on unsupervised learning is developed to form clusters of channels with similar electronic pulse signals, which allows the use of the same neural network for all channels in one cluster. This method reduces the number of needed neural networks to about 100 making it possible to cover the full detector with these advanced algorithms.

Keywords: machine learning, liquid argon calorimeter, LHC, ATLAS, FPGA

Remerciements

This work received support from the French government under the France 2030 investment plan, as part of the Excellence Initiative of Aix-Marseille University - A*MIDEX (AMX-19-IET-008 - IPhU).

Contents

List of Figures

List of Tables

Synthèse en français

Introduction

Le Modèle Standard de la physique des particules est la théorie la plus aboutie pour expliquer les particules élémentaires et leurs interactions. Il a été complété après la découverte du boson de Higgs au Grand Collisionneur de Hadrons (LHC) en 2012 par les expériences ATLAS et CMS. Malgré son succès, le Modèle Standard est réputé être un tremplin vers une théorie plus complète décrivant les phénomènes qu'il ne parvient pas à expliquer, tels que la matière noire, l'énergie noire et l'asymétrie matière-antimatière. Les données recueillies jusqu'à présent au LHC ont été suffisantes pour découvrir le boson de Higgs et de mesurer un grand nombre de ses propriétés, mais toutes les recherches de physique au-delà du Modèle Standard ont été infructueuses. Toutefois, l'essentiel des données des expériences au LHC sont encore à venir.

Le LHC à haute luminosité (HL-LHC) est la forme ultime de l'accélérateur qui vise à augmenter sa luminosité instantanée d'un facteur de 5 à 7 par rapport à sa luminosité nominale pour atteindre $5-7\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$, ce qui se traduit par 140 à 200 interactions proton-proton par croisement de paquets. Le HL-LHC devrait fonctionner jusqu'en 2040 et recueillir dix fois plus de données que les cycles précédents du LHC réunis. Des mises à niveau sont nécessaires pour que les détecteurs puissent faire face à l'augmentation de la luminosité. Elles sont prévues pour 2026-2028 et cette phase est connue sous le nom de mise à niveau de phase II. Le calorimètre à argon liquide (LAr) d'ATLAS mesure l'énergie des particules sensibles à l'interaction électromagnétique produites dans les collisions proton-proton au LHC. L'électronique de lecture du calorimètre LAr sera remplacée pour le préparer au HL-LHC. Cela lui permettra de fonctionner à un taux de déclenchement plus élevé et d'avoir une granularité accrue au niveau du déclenchement. L'énergie déposée dans le calorimètre LAr est calculée à partir du signal d'impulsion électronique à l'aide d'un algorithme de filtrage optimal. L'énergie est reconstruite en temps réel à l'aide de cartes électroniques personnalisées basées sur des circuits logique programmables (FPGA). Les FPGAs ont été choisis en raison de leur capacité à traiter de grandes quantités de données avec une faible latence, ce qui est une exigence du système de déclenchement d'ATLAS. L'augmentation de la luminosité du LHC entraînera un taux élevé de collisions multiples simultanées entre protons et protons (empilement), ce qui se traduira par une dégradation significative de la résolution de l'énergie calculée par l'algorithme de filtrage optimal. Le calcul de l'énergie avec une grande précision est de la plus haute importance pour atteindre les objectifs de physique de l'expérience ATLAS au HL-LHC.

Les avancées récentes en matière d'apprentissage profond couplées à l'augmentation de la capacité de calcul des FPGA font des algorithmes d'apprentissage profond des outils prometteurs pour remplacer les algorithmes de filtrage optimal existants. Dans cette thèse, des réseaux de neurones récurrents (RNN) sont développés pour calculer l'énergie déposée dans le calorimètre LAr. Des réseaux récurrents à mémoire court et long terme (LSTM) et des réseaux de neurones simples sont étudiés. Les paramètres de ces réseaux de neurones sont étudiés en détail afin d'optimiser les performances et de trouver des architectures de réseau qui peuvent s'adapter aux FPGAs. Les réseaux développés se révèlent plus performants que les algorithmes de filtrage optimaux. Les modèles sont optimisés davantage pour être déployés sur les FPGA par des méthodes de quantification et de compression qui permettent de réduire la consommation de ressources avec un effet minimal sur les performances.

Le calorimètre LAr est composé de 182000 canaux individuels pour lesquels nous devons calculer les énergies déposées. L'entraînement de 182000 réseaux neuronaux différents n'est pas réalisable en pratique. Une nouvelle méthode basée sur l'apprentissage non supervisé pour former des groupes de canaux de calorimètre avec des signaux d'impulsion électroniques similaires est developpée. Cela permet l'utilisation du même réseau de neurones pour tous les canaux dans un même groupe. Cette méthode réduit le nombre de réseaux de neurones nécessaires à une centaine environ, ce qui permet d'exploiter l'ensemble du détecteur avec ces algorithmes avancés.

Algorithmes actuels de reconstruction de l'énergie

Le calorimètre à argon liquide détecte les particules qui ionisent l'argon, qui joue le rôle de millieu actif. Les électrons libres sont collectés grâce à des électrodes à haute tension. La réponse du détecteur est une impulsion triangulaire, comme le montre la figure 0.1. Cette réponse a une montée rapide et une descente qui dure jusqu'à 500ns. Un filtre analogique $CR-RC^2$ est utilisé pour filtrer le signal. La réponse après application du filtre est une forme d'impulsion bipolaire à partir de laquelle il est possible d'extraire l'amplitude et le temps du dépôt. Cette réponse très stable ne varie qu'au niveau de sa hauteur de crête, qui est proportionnelle à l'énergie déposée et au temps du dépôt. Ces deux facteurs, la hauteur et le temps du pic, sont les objectifs de la reconstruction dans le LAr. La reconstruction se fait à partir de plusieurs points d'échantillonnage sur le signal comme le montre la figure 0.1.

Filtrage optimal

L'algorithme de filtrage optimal est utilisé dans le calorimètre LAr pour extraire l'amplitude et la position dans le temps d'un signal bruyant afin de minimiser les effets de l'empilement et d'autres sources de bruit telles que le bruit thermique. L'amplitude de l'impulsion A est reconstruite avec la formule 0.1 et le déphasage (temps relatif du pic) avec la formule 0.2. Ces formules sont des sommes pondérées des échantillons de signaux numérisés ($n = 5$ dans les études de phase II) en utilisant les coefficients de filtrage a_i pour l'amplitude de l'impulsion et b_i pour le temps relatif τ mesurant le

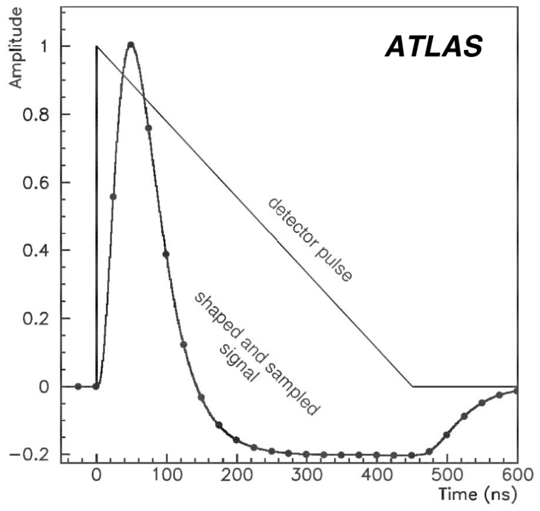


Figure 0.1: Forme d'impulsion électronique dans le LAr avec les échantillons à 40 MHz.

déphasage entre le moment prévu et le moment réel de l'impulsion.

$$A = \sum_{i=1}^{n} a_i \cdot s_i \tag{0.1}$$

$$A\tau = \sum_{i=1}^{n} b_i \cdot s_i \tag{0.2}$$

Mesure continue de l'énergie avec les algorithmes de filtrage optimal

Pour la mesure continue de l'énergie, un détecteur de maximum est utilisé sur la sortie du filtrage optimal pour attribuer l'énergie E à un croisement de paquets spécifique. La condition pour attribuer l'énergie à un croisement de paquet est que celle ci soit l'énergie maximale par rapport aux croisements précédents et suivants, et qu'elle soit supérieure à un seuil défini E_{thr}. Dans le cas contraire, l'énergie attribuée à ce croisement est nulle. La condition de recherche du maximum est formulée dans l'équation 0.3. La combinaison de l'algorithme de filtrage optimal avec la recherche du maximum est appelée OFMax.

$$E = \begin{cases} E_{i-1} & \text{if} \quad (E_{i-1} > E_i) \wedge (E_{i-1} > E_{i-2}) \wedge (E_{i-1} > E_{thr}) \\ 0 & \text{sinon} \end{cases} \tag{0.3}$$

L'augmentation de la luminosité du HL-LHC introduira des conditions d'empilement qui réduiront considérablement les performances de la technique de filtrage optimal et nécessiteront de nouvelles techniques de traitement du signal. Cette thèse évalue la possibilité de reconstruire l'énergie avec des réseaux de neurones récurrents qui seront employés sur les FPGAs des nouvelles cartes de traitement des données du calorimètre LAr.

Reconstruction de l'énergie avec des réseaux de neurones récurrents en fenêtres coulissantes

L'application des RNNs est effectué avec une architecture comportant des fenêtres coulissantes des échantillons. La séquence d'échantillons complète est divisé en séquences plus courtes qui se chevauchent; chacune des séquences correspond à un dépôt d'énergie. L'énergie cible pour chaque sous-séquence peut être choisie de manière à ce que la séquence contienne des informations à la fois avant le dépôt, pour corriger les effets d'empilement, et des échantillons après le dépôt pour la reconstruction de l'amplitude du signal. La position de l'énergie cible par rapport à la sous-séquence doit être choisie pour obtenir la performance optimale des réseaux, tout en gardant à l'esprit que chaque échantillon ajouté dans la séquence augmente l'utilisation des ressources dans les FPGAs. De plus, les échantillons après le dépôt ajouteront de la latence au temps de calcul de l'énergie.

L'architecture choisie des réseaux consiste en une couche RNN suivie d'un seul neurone dans la couche de sortie. Quatre échantillons sont utilisés après le dépôt d'énergie en plus de plusieurs échantillons dans le passé. La figure 0.2 montre un exemple de la division de la chaîne complète en fenêtres coulissante d'une longueur de cinq échantillons.

Performance de la reconstruction de l'énergie

La performance des RNNs pour la reconstruction des énergies est affectée par plusieurs facteurs. L'un de ces facteurs est la taille du réseau RNN en termes de nombre d'unités. La taille du réseau détermine la capacité du réseau à apprendre des modèles complexes et peut avoir un impact significatif sur sa capacité à se généraliser à des données non vues. Un autre facteur important est la taille de la fenêtre coulissante utilisée, qui affecte à la fois la correction de l'empilement et la reconstruction de l'amplitude. L'optimisation de ces facteurs est cruciale pour obtenir les meilleures performances possibles en matière de reconstruction de l'énergie. Les réseaux évalués sont le RNN simple et le LSTM qui est plus complexe.

Résumé des performances des réseaux en fonction de la longueur de la séquence

Pour répondre aux exigences strictes en termes de latence et d'occupation du FPGA, il est essentiel que le réseau soit aussi petit que possible. La figure 0.3 présente les performances des RNN simples et des LSTM en fonction de la longueur de la séquence. Comme prévu, la séquence la plus longue (30) présente les meilleures performances, mais elle nécessite 6 fois plus de calculs que la séquence la plus courte (5). Tous les réseaux LSTM bénéficient d'une longueur de séquence plus importante, mais seuls les

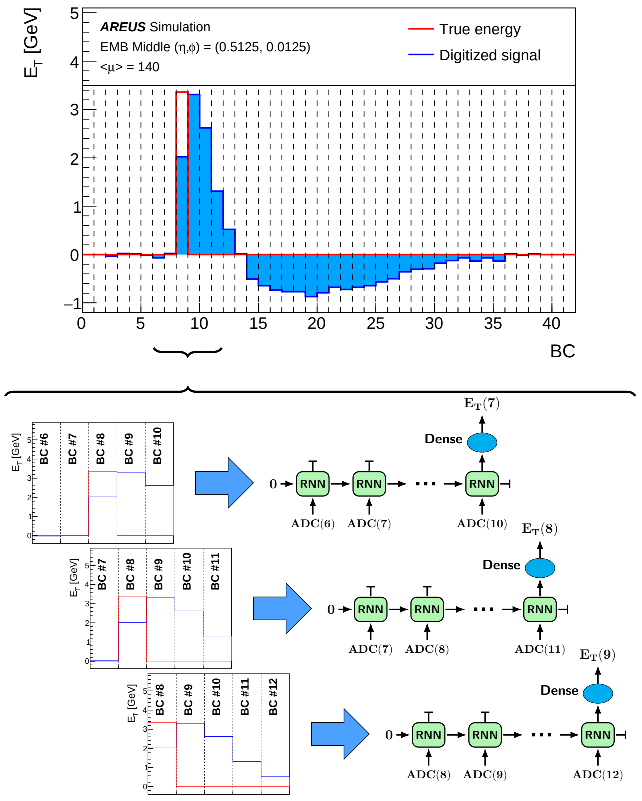


Figure 0.2: Application du RNN avec des fenêtres coulissantes, où une fenêtre de taille fixe est déplacée sur la séquence des échantillons pour traiter les données en segments plus petits qui se chevauchent.

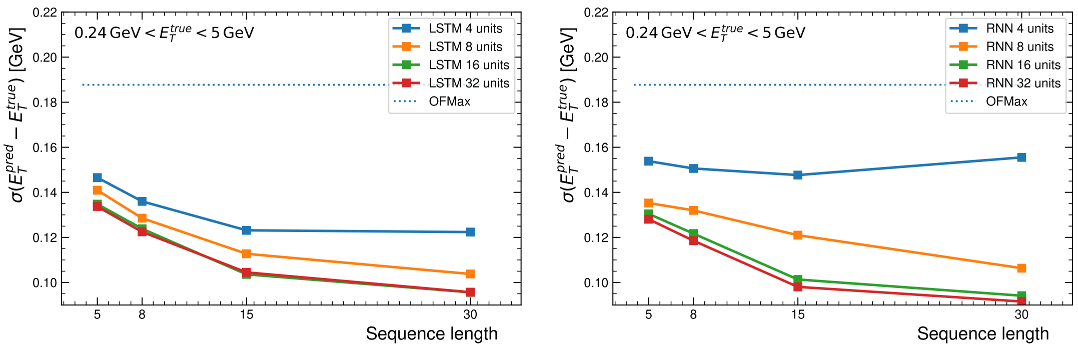


Figure 0.3: Résolution de l'énergie transversale en fonction de la longueur de la séquence pour les LSTMs (à gauche) et les RNNs simples (à droite) avec différents nombres d'unités internes.

trois plus grands RNNs bénéficient du même avantage. En fait, le RNN avec seulement 4 unités n'est pas en mesure de profiter de la longueur de séquence plus importante en raison de la taille réduite de son réseau neuronal interne. Pour les deux types de réseaux, on peut observer une structure en coude à une longueur de séquence de 15, où le doublement de la longueur de séquence à 30 n'apporte pas d'amélioration significative à la performance.

Résumé de la performance des réseaux en fonction du nombre d'unités

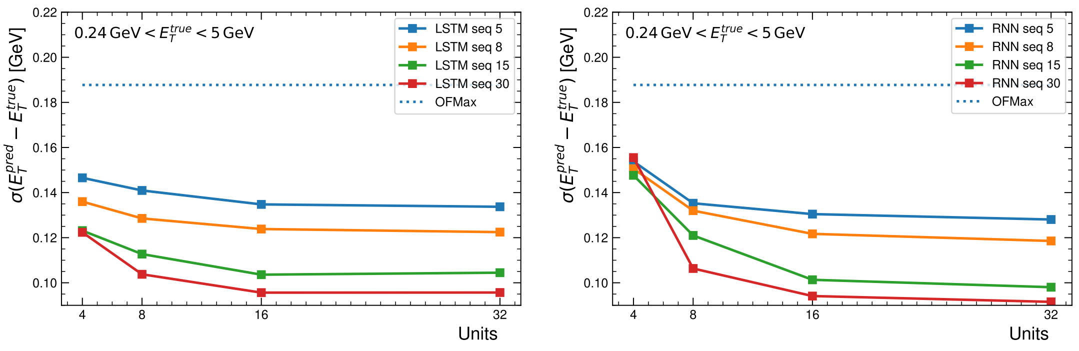


Figure 0.4: Résolution de l'énergie transversale en fonction du nombre d'unités des LSTMs (à gauche) et des RNNs simples (à droite).

Le nombre de paramètres augmente quadratiquement avec le nombre d'unités. Alors, il est essentiel de maintenir le nombre d'unités aussi bas que possible. La figure 0.4 montre les performances en fonction du nombre d'unités du RNN pour différentes longueurs de séquences, à la fois pour le LSTM et le RNN simple. L'ajout d'unités améliore les performances, mais l'effet est faible au-delà de 16 unités.

Optimisation des RNNs pour le déploiement sur FPGA

L'implémentation des RNNs discutés ci-dessus sur FPGA a déjà été effectuée et les résultats sont publiés dans [1] et [2]. Ces implémentations ont utilisé des réseaux de neurones entraînés avec des nombres à virgule flottante et implémentés en utilisant des nombres virgule fixe dans le micrologiciel, ce qui correspond à la quantification des opérations arithmétiques du réseau. La conversion de la virgule flottante à la virgule fixe a entraîné une perte de précision, mais le nombre de bits dans la représentation en virgule fixe a été augmenté pour réduire cet effet. Dans ce chapitre, l'entraînement prenant en compte la quantification sera étudié pour effectuer l'apprentissage des réseaux neuronaux avec une arithmétique en virgule fixe. Cela permet de réduire encore le nombre de bits nécessaires, et donc les ressources des FPGA, sans perte de précision. D'autres méthodes visant à réduire les ressources, telles que l'élagage des réseaux et une initialisation des RNN basée sur un réseau avec une couche dense, sont également étudiées.

L'entraînement conscient de la quantification

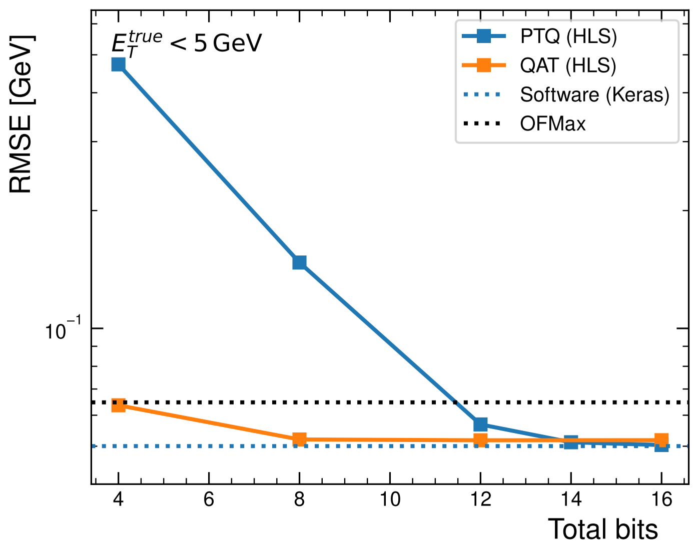


Figure 0.5: Erreur quadratique moyenne (RMSE) en fonction du nombre de bits pour un RNN simple avec 8 unités et une longueur de séquence de 5 quantifiée avec les méthodes PTQ et QAT, comparée aux résultats d'OFMax et du réseau neuronal avec une précision en virgule flottante dans le logiciel. La partie entière est de 3 bits pour PTQ et de 1 bit pour QAT.

L'entraînement conscient de la quantification (QAT) est une méthode d'apprentissage des réseaux de neurones qui atténue, pendant la durée de la formation, la perte de performance causée par l'utilisation de représentations numériques à virgule fixe avec un faible nombre de bits. Cette méthode diffère de la quantification post-entraînement (PTQ) où le nombre de bits approprié est déterminée après l'entraînement du réseau neuronal avec la précision de la virgule flottante. Lors de la PTQ, l'implémentation FPGA a besoin d'un nombre de bits suffisant pour reproduire les opérations mathématiques sans perte significative de performance, ce qui nécessite généralement un nombre de bits élevé. Ce problème peut être évité par le QAT où le nombre de bit est déterminé pendant l'entraînement.

Le RNN simple avec 8 unités et une longueur de séquence de 5 est utilisé pour étudier l'apprentissage quantifié. Dans des travaux antérieurs [2], PTQ a été utilisé pour trouver le nombre de bits approprié pour les poids de ce réseau. Un minimum de 14 bits au total, dont trois bits réservés à la partie entière, s'est avéré nécessaire. Un nombre de bits inférieur entraîne une perte de performance inacceptable.

La figure 0.5 montre la résolution en énergie en fonction du nombre de bits total pour différents entraînements de RNN et pour OFMax. La résolution pour les réseaux quantifiés est évaluée à l'aide d'une simulation d'une implémentation de micrologiciel réalisée avec le langage High Level Synthesis (HLS) comme expliqué dans [1]. On peut constater la perte de performance du PTQ pour une largeur de bit inférieure à 14. Un QAT est réalisé pour le réseau RNN simple à l'aide de QKeras [3]. La QAT atteint la précision totale en virgule flottante implémentée dans le logiciel à partir d'une largeur de bit de 8. Cela fait de la QAT une méthode appropriée pour réduire considérablement les ressources nécessaires dans le FPGA.

Processus de regroupement

Pour utiliser les réseaux neuronaux pour la reconstruction de l'énergie dans toutes les cellules du détecteur, il est essentiel de disposer d'un ensemble de RNNs performants pour chaque cellule. Les algorithmes traditionnels et les RNN sont tous deux sensibles aux changements dans la forme des impulsions d'une cellule à l'autre du calorimètre. Il serait possible d'entraîner un réseau pour l'ensemble des 182 468 cellules de manière indépendante, mais cela engendrerait un temps d'entraînement insoutenable. Pour conserver des performances élevées dans toutes les cellules et disposer d'un nombre limité de RNNs, il est possible d'utiliser l'apprentissage non supervisé pour identifier les régions du détecteur qui ont une réponse similaire. En regroupant les impulsions sur la base de leur similarité, les algorithmes d'apprentissage non supervisé peuvent identifier des groupes de cellules du détecteur qui ont des propriétés similaires. Un groupe identifié utilisera alors le même RNN, ce qui réduit considérablement le temps d'apprentissage nécessaire. Outre l'identification des cellules ayant une réponse similaire, il est également possible de détecter les cellules ayant une réponse anormale.

Les impulsions des cellules actuelles du calorimètre sont mesurées avec 768 échantillons qui peuvent être représentés par 768 dimensions. Le regroupement procède d'abord par une réduction de la dimensionnalité avec t-SNE à deux dimensions puis le regroupement est effectué dans ces deux dimensions avec DBSCAN.

La figure 0.6 montre les performances des RNN avec différents entraînements pour une cellule de la couche 3 du tonneau (EMB) du calorimètre LAr. Quatre réseaux neuronaux sont comparés : le premier est entraîné à l'aide des données du même groupe de cette cellule. Dans ce cas, la forme de l'impulsion de la cellule au centre du groupe est utilisée pour la simulation des données. Le deuxième réseau est formé à partir des données de la même cellule. Le troisième réseau est entraîné sur une autre cellule choisie au hasard dans la couche 3 de l'EMB. Le dernier réseau est entraîné sur des données simulées avec un mélange de formes d'impulsions provenant de tous les groupes de la couche 3 de l'EMB.

On constate que les réseaux entraînés sur la même cellule et sur la cellule au centre du même groupe donnent des résultats similaires. Cela confirme que la procédure est capable de regrouper les cellules ayant des formes d'impulsion similaires. Le réseau entraîné sur un autre groupe présente des performances très médiocres. L'entraînement sur des données contenant un mélange de formes d'impulsions provenant de tous les amas ne permet pas de retrouver les mêmes performances que celui effectué sur la même cellule. Ce résultat démontre la nécessité et l'efficacité de la procédure de regroupement développée durant cette thèse.

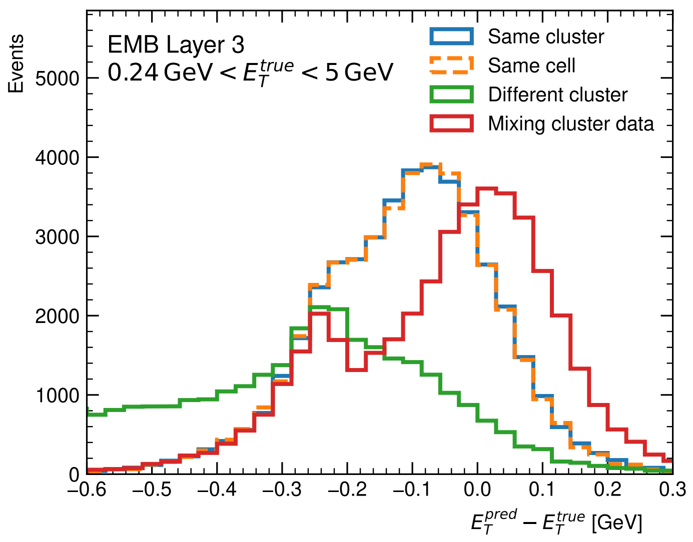


Figure 0.6: $E_T^{pred} - E_T^{true}$ des RNNs pour une cellule dans la couche 3 du EMB. En bleu, le RNN est entraîné sur une autre cellule du même groupe. En orange, le RNN est entraîné sur la même cellule que celle utilisée pour le test. En vert, le réseau est entraîné sur une cellule d'un autre groupe. En rouge, le RNN est entraîné sur un ensemble de données combinant les données de tous les groupes.

Conclusion

La luminosité élevée sans précédent du HL-LHC entraîne une réduction de la résolution de l'énergie reconstruite dans le calorimètre LAr d'ATLAS par l'algorithme de filtrage optimal. Pour atteindre les objectifs de physique de l'expérience ATLAS, des méthodes nouvelles et améliorées de reconstruction de l'énergie sont nécessaires. Le système de lecture du calorimètre LAr sera remplacé dans le cadre de la mise à niveau de la phase II en 2026-2028. La nouvelle électronique sera basée sur des FPGAs de pointe avec une capacité de calcul accrue permettant d'effectuer la reconstruction de l'énergie à l'aide de réseaux de neurones.

Cette thèse évalue les RNNs en tant que candidats pour remplacer l'algorithme de filtrage optimal. Les architectures RNN simple et LSTM ont été développées et se sont avérées plus performantes que l'algorithme de filtrage optimal dans les conditions du HL-LHC. En particulier, ces RNN sont conçus pour calculer avec précision l'énergie des impulsions qui se chevauchent, un domaine dans lequel l'algorithme de filtrage optimal est peu performant. Les RNNs sont capables d'évaluer les corrections nécessaires en cas de distorsion des impulsions due au chevauchement en incorporant des informations sur les événements antérieurs au dépôt d'énergie. En outre, les RNNs se révèlent résistants aux variations de l'empilement moyen au cours des runs du LHC : Les RNNs entraînés sur des échantillons dont l'empilement moyen se situe entre 100 et 200 ont de très bonnes performances sur n'importe quel échantillon dont l'empilement moyen se situe dans la même fourchette. Enfin, des RNNs capables de calculer simultanément l'énergie et le déphasage dans le temps du pic d'impulsion sont développés. Le déphasage est important pour attribuer l'énergie déposée au croisement de faisceaux correspondant et pour rechercher les particules à longue durée de vie. Les RNNs sont plus performants que l'algorithme de filtrage optimal pour la reconstruction de l'énergie et du déphasage en temps.

Le déploiement des RNNs sur les FPGAs requiert des considérations particulières. Afin de réduire les ressources dans les FPGAs, l'arithmétique en virgule fixe avec le plus petit nombre de bits possible est nécessaire. L'entraînement du réseau avec des poids quantifiés permet de réduire le nombre de bits d'un facteur deux tout en conservant une précision similaire à celle de la virgule flottante. Cela conduit à une réduction drastique des ressources nécessaires dans les FPGA. Une autre façon de réduire les ressources consiste à élaguer les réseaux de neurones en supprimant les poids non significatifs. Toutefois, il est démontré qu'il est plus efficace de commencer par de petits RNNs sans élagage que de former de grands RNNs et de les élaguer ensuite. L'élagage n'est donc pas envisagé pour les RNNs utilisés pour le calcul de l'énergie dans le calorimètre LAr. Enfin, une nouvelle architecture utilisant une couche dense, qui sonde les informations passées, pour initialiser le RNN est développée. Cette architecture est capable de reproduire les performances des RNN avec de longues séquences tout en nécessitant 5 fois moins de puissance de calcul. Tous ces développements permettent d'implémenter un RNN efficace sur des FPGAs.

Pour utiliser les RNNs pour le calorimètre LAr complet, 182000 réseaux neuronaux sont nécessaires. L'entraînement d'un si grand nombre de RNNs est pratiquement impossible. Pour éviter ce problème, l'apprentissage non supervisé est utilisé pour regrouper automatiquement les cellules du calorimètre sur la base de similitudes dans leur réponse électronique. Cette méthode réduit le nombre de réseaux de neurones récurrents requis à 100 pour les parties électromagnétiques du détecteur, ce qui rend l'utilisation des RNN plus réaliste pour le calorimètre LAr complet.

En résumé, les RNNs se révèlent être d'excellents candidats pour reconstruire l'énergie déposée dans le calorimètre LAr durant le HL-LHC. Ils sont plus performants que l'algorithme de filtrage optimal actuel et peuvent être mis en œuvre sur des FPGAs. Des travaux supplémentaires sont encore nécessaires pour optimiser davantage les réseaux neuronaux et pour réaliser l'implémentation finale dans le microprogramme FPGA. En outre, des méthodes d'étalonnage de la réponse des RNNs avec des données réelles doivent encore être mises au point pour tenir compte des différences entre les données de simulation utilisées pour l'entraînement et les données réelles du HL-LHC.

Introduction

The standard model of particle physics is the most successful theory to explain elementary particles and their interactions. It was completed after the discovery of the Higgs boson at the Large Hadron Collider (LHC) in 2012 by ATLAS and CMS experiments. Despite its success the standard model is thought to be a stepping stone to a more comprehensive theory that explains phenomena that the standard model fails to explain such as dark matter, dark energy and matter-antimatter asymmetry. The data collected until recently by the ATLAS and CMS collaborations allowed to discover the Higgs boson and study many of its properties, however, all searches for new physics beyond the standard model have been unsuccessful. The LHC will collect more data with increased precision in the HL-LHC phase which offers the opportunity to further study the Higgs boson and discover rare phenomena not described by the standard model.

The High-Luminosity LHC (HL-LHC) is the last phase in the LHC lifetime that aims to increase its instantaneous luminosity by a factor of 5-7 with respect to its design luminosity to $5-7\times10^{34}\mathrm{cm}^{-2}\mathrm{s}^{-1}$. This translates to 140 to 200 proton-proton interactions every 25 ns. The HL-LHC is scheduled to be running until 2040 and to gather 10 times more data than the previous LHC runs combined. Upgrades are required for the detectors to be able to cope with the increased luminosity. The upgrades are planned during the so called Phase-II upgrade that will happen in 2026-2028.

The ATLAS liquid argon (LAr) calorimeter measures mainly the energies of electromagnetically interacting particles produced in proton-proton collisions at the LHC. The readout electronics of the LAr calorimeter will be replaced to prepare it for the HL-LHC era. This will allow it to run at a higher trigger rate and have increased granularity at the trigger level. The energy deposited in the LAr calorimeter is reconstructed out of the electronic pulse signal using the optimal filtering algorithm. The energy is reconstructed in real-time using custom electronic boards based on Field Programmable Gate Arrays (FPGAs). FPGAs are chosen due to their ability to process large amounts of data with low latency which is a requirement of the ATLAS trigger system. The increased LHC luminosity will lead to a high rate of simultaneous multiple proton-proton collisions (pileup) that results in a significant degradation of the energy resolution computed by the optimal filtering algorithm. Computing the energy with high precision is of utmost importance to achieve the physics goals of the ATLAS experiment at the HL-LHC.

Recent advances in deep learning coupled with the increased computing capacity of FPGAs makes deep learning algorithms promising tools to replace the existing optimal filtering algorithms. In this dissertation, recurrent neural networks (RNNs) are developed to compute the energy deposited in the LAr calorimeter. After an overview of particle physics in chapter 1 and of neural networks in chapter 2, an investigation of the usage of recurrent neural networks for the energy reconstruction in the LAr calorimeter is presented in chapter 3. The parameters of these neural networks are studied in detail to optimize the performance while focusing on architectures that can fit into FPGAs. The developed networks are shown to outperform the optimal filtering algorithms. In chapter 4 the models are further optimized to be deployed on FPGAs by quantization and compression methods which are shown to reduce the resource consumption with minimal effect on the performance. New architectures with reduced computational needs are also presented.

The LAr calorimeter is composed of 182000 individual channels for which we need to compute the deposited energies. Training 182000 different neural networks is not practically feasible. Chapter 5 introduces a new method based on unsupervised learning to form clusters of calorimeter channels with similar electronic pulse signals, which allows the use of the same neural network for all channels in one cluster. This method reduces the number of needed neural networks to about 100 making it possible to cover the full detector with these advanced algorithms.

1 The ATLAS Experiment

Summary

1.1 The Standard Model of Particle Physics

The Standard Model of particle physics is the most successful theory to explain the elementary particles and their interactions [4]. Despite its ability to explain a broad range of phenomena, it has limitations which suggest an underlying and more fundamental theory. Precision measurements of standard model processes and the searches for physics beyond the standard model (BSM) are required to probe for new physics. These include searches for dark matter, super symmetry and matter-antimatter asymmetry, which will be the key areas of research in the coming decades.

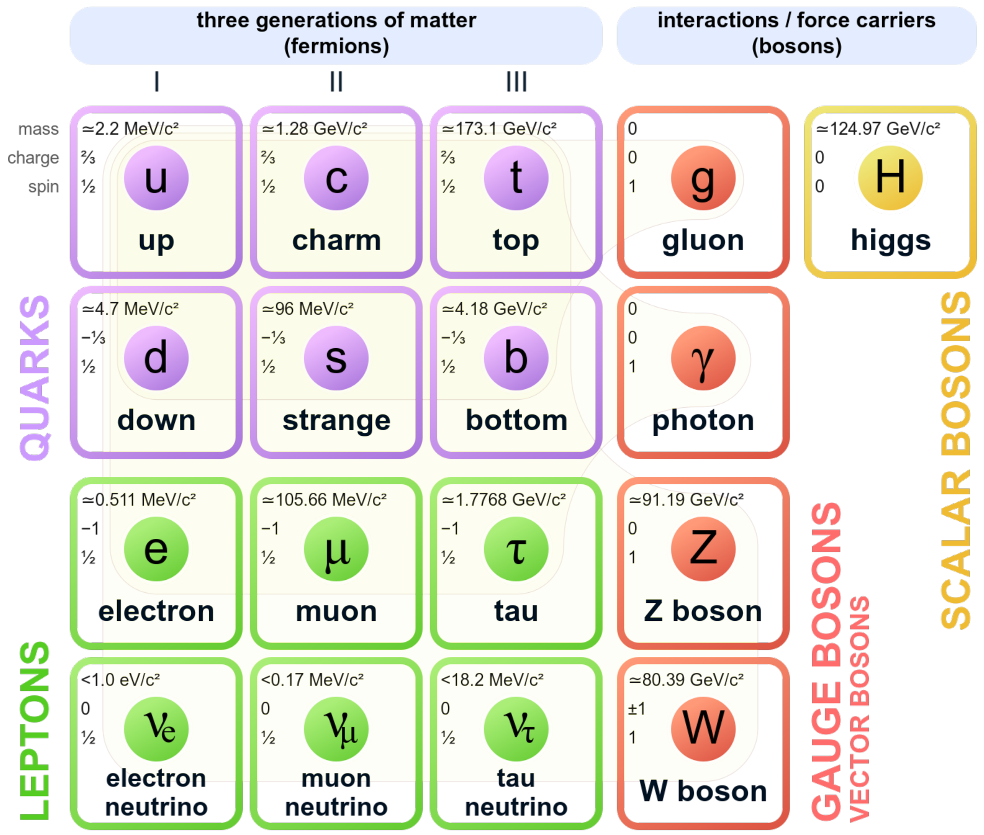

three generations of matter (fermions)
interactions / force carriers (bosons)
I
II
III
mass
charge
spin
≃2.2 MeV/c² ⅔ ½ u up
≃1.28 GeV/c² ⅔ ½ c charm
≃173.1 GeV/c² ⅔ ½ t top
0 0 1 g gluon
≃124.97 GeV/c² 0 0 H higgs
≃4.7 MeV/c² −⅓ ½ d down
≃96 MeV/c² −⅓ ½ s strange
≃4.18 GeV/c² −⅓ ½ b bottom
0 0 1 γ photon
≃0.511 MeV/c² −1 ½ e electron
≃105.66 MeV/c² −1 ½ μ muon
≃1.7768 GeV/c² −1 ½ τ tau
≃91.19 GeV/c² 0 1 Z Z boson
<1.0 eV/c² 0 ½ νe electron neutrino
<0.17 MeV/c² 0 ½ νμ muon neutrino
<18.2 MeV/c² 0 ½ ντ tau neutrino
≃80.39 GeV/c² ±1 1 W W boson
QUARKS
LEPTONS
GAUGE BOSONS
VECTOR BOSONS
SCALAR BOSONS

Figure 1.1: Standard model of elementary particles [5].

1.1.1 Fermions and Matter

The standard model has 12 fermions which are divided into quarks and leptons depending on whether they interact strongly or not [4]. The quarks interact strongly while leptons do not.

Both quarks and leptons are divided further into three generations where the generations have similar properties but different scales of mass. The particles in the first generation are the only ones to be stable. These stable particles constitute the material of our surroundings by forming composite particles: the electron and the up and down quarks [4]. The composite particles are called hadrons. For example the nuclei of atoms are formed by hadrons. Protons have two up quarks and one down quark (uud) and neutrons have two down quarks and one up quark (udd).

1.1.2 Interactions

Out of the four fundamental forces, three are explained by the standard model as an exchange of gauge bosons between elementary particles [4].

Gravitational interaction is not explained by the standard model. It acts between a pair of bodies that have non-zero mass. Gravity is the dominant force at macroscopic scales and it is explained by the general theory of relativity. At present, only extensions of the standard model include gravitational interaction but no evidence of a particle responsible for it has been found.

Electromagnetic interaction is propagated by photons (γ) and acts between pairs of electrically charged particles. It affects all matter except for the neutrinos.

Weak interaction is propagated by the W and Z bosons. It affects particles with weak charge. The weak force is for instance responsible for the beta decay.

Strong interaction is propagated by gluons (g) and is responsible for binding the atomic nuclei together. Only quarks are sensitive to the strong interaction.

1.1.3 The Higgs Boson

The standard model also includes the Higgs mechanism that is responsible for the mass of elementary particles. Particles are intrinsically massless and their interaction with the Higgs field is what gives them mass. An excitation of the Higgs field produces a new particle that is called the Higgs boson [4]. The Higgs boson was the last particle to be observed and its discovery was announced in 2012 by the ATLAS [6] and CMS [7] experiments at the LHC.

1.2 CERN

The European Organization of Nuclear Research (CERN) was established in 1954 as an European collaboration for fundamental science. Since then it has been the home of multiple physics experiments and evolved into an international collaboration.

1.2.1 The Large Hadron Collider

The LHC [8] is the largest particle accelerator and collider in the world, located at the CERN accelerator complex, seen in figure 1.2. It is 26.7 km long and has two counter-rotating proton beams bent by magnetic dipoles with opposite fields. The beams intersect at interaction points where the detectors are located. The LHC was built to accelerate and to collide particles, namely proton-proton collisions at center-of-mass energy of 14 TeV as well as to collide ions. Four main detectors in the LHC are ALICE, ATLAS, CMS and LHCb. Of these ATLAS and CMS are general purpose detectors while ALICE and LHCb are more specialized detectors.

The particles used in the collisions in the LHC are supplied to it via an injector chain [8]. The injector chain for protons consists of several steps: Linac4 - Proton Synchrotron Booster (PSB) - Proton Synchrotron (PS) - Super Proton Synchrotron (SPS). A radio frequency (RF) system is used to accelerate the particles and to organize them into bunches instead of a continuous stream. These bunches are colliding at a frequency of 40 MHz, every 25 ns, in what are called bunch crossings (BC). Each bunch consists of a nominal value of $O(10^{11})$ protons. The full accelerator is able to hold up to 2808 bunches, but usually not all of them contain protons. Different filling schemes are called bunch trains and the densest structure used is 72 consecutive bunches followed by 8 empty bunches.

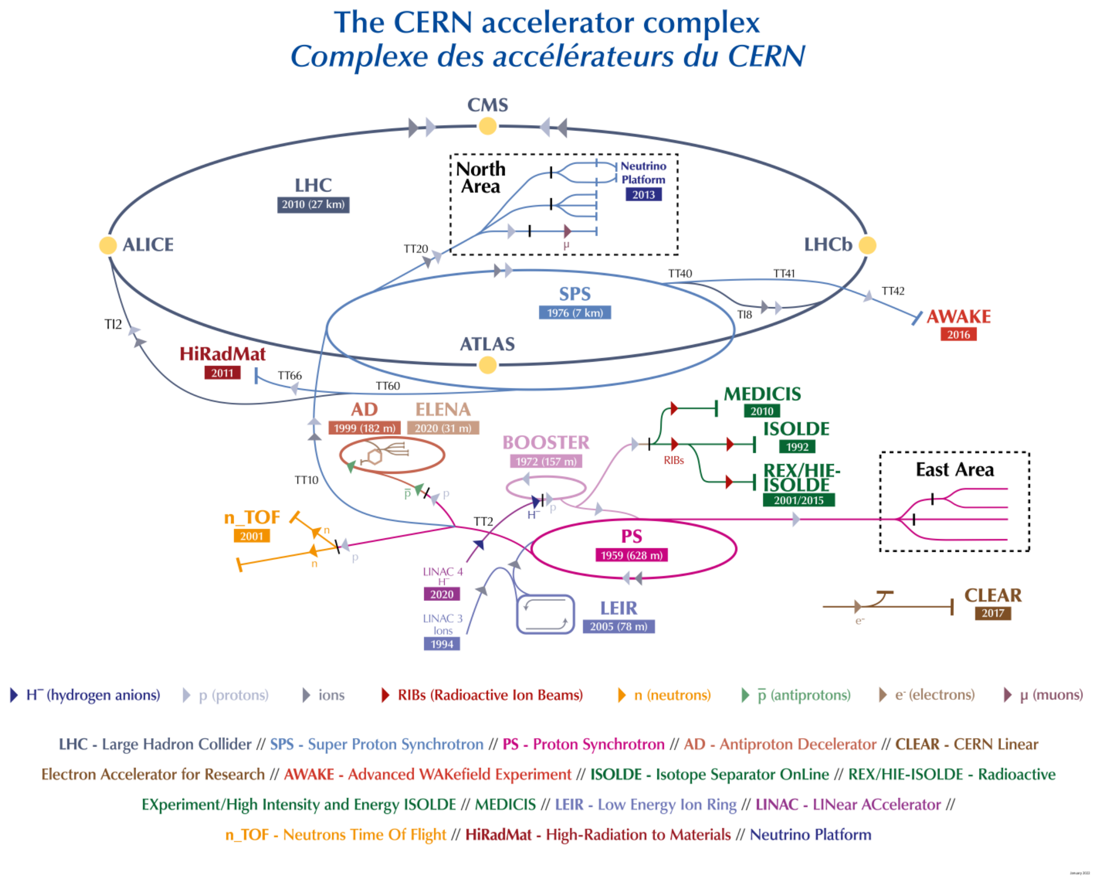

The CERN accelerator complex
Complexe des accélérateurs du CERN
CMS
North Area
Neutrino Platform
2013
LHC
2010 (27 km)
ALICE
LHCb
TT20
TT40
TT41
TT42
AWAKE
2016
SPS
1976 (7 km)
TI8
TI2
ATLAS
HiRadMat
2011
TT66
TT60
AD
1999 (182 m)
ELENA
2020 (31 m)
BOOSTER
1972 (157 m)
MEDICIS
2010
ISOLDE
1992
REX/HIE-ISOLDE
2001/2015
RIBs
East Area
TT10
n_TOF
2001
TT2
PS
1959 (628 m)
LINAC 4
H⁻
2020
LINAC 3
Ions
1994
LEIR
2005 (78 m)
CLEAR
2017
H⁻ (hydrogen anions)
p (protons)
ions
RIBs (Radioactive Ion Beams)
n (neutrons)
p̄ (antiprotons)
e⁻ (electrons)
μ (muons)
LHC - Large Hadron Collider // SPS - Super Proton Synchrotron // PS - Proton Synchrotron // AD - Antiproton Decelerator // CLEAR - CERN Linear Electron Accelerator for Research // AWAKE - Advanced WAKefield Experiment // ISOLDE - Isotope Separator OnLine // REX/HIE-ISOLDE - Radioactive EXperiment/High Intensity and Energy ISOLDE // MEDICIS // LEIR - Low Energy Ion Ring // LINAC - LINear ACcelerator // n_TOF - Neutrons Time Of Flight // HiRadMat - High-Radiation to Materials // Neutrino Platform

Figure 1.2: The CERN accelerator complex [9].

1.2.2 Luminosity

The rate of collisions in a particle accelerator is given by its luminosity [10]. The luminosity of a design with two counter-rotating beams with bunched particles, such as the LHC, can be approximated by the formula in equation 1.1. The luminosity depends on the beam parameters where N_b is the number of particles per bunch, n_b the number of bunches per beam, f_{rev} the revolution frequency, γ_r the relativistic gamma factor, ε_n the normalized transverse beam emittance, $\beta*$ the beta function at the collision point and F the geometric luminosity reduction factor due to the crossing angle at the interaction point.

$$\mathscr{L} = \frac{N_b^2 n_b f_{rev} \gamma_r}{4\pi\varepsilon_n \beta*} F \tag{1.1}$$

The unit for collisions happening over a period of time is an inverse femtobarn (fb^{-1}) which corresponds to approximately 10^{12} proton-proton collisions, this is also commonly referred to as integrated luminosity.

The design luminosity of the LHC (10^{34} cm$^{-2}s^{-1}$) was surpassed in 2016. Figure 1.3

shows the luminosity and the integrated luminosity of the ATLAS experiment over different runs as well as the average number of proton-proton interactions per bunch crossing, this is referred to as pileup, and is denoted by $\langle\mu\rangle$. Figure 1.4 shows the cumulative luminosity delivered to ATLAS for p-p collisions over time. The instantaneous luminosity for each run has been increasing and the preliminary average pileup for Run-3 is $\langle\mu\rangle = 42.5$.

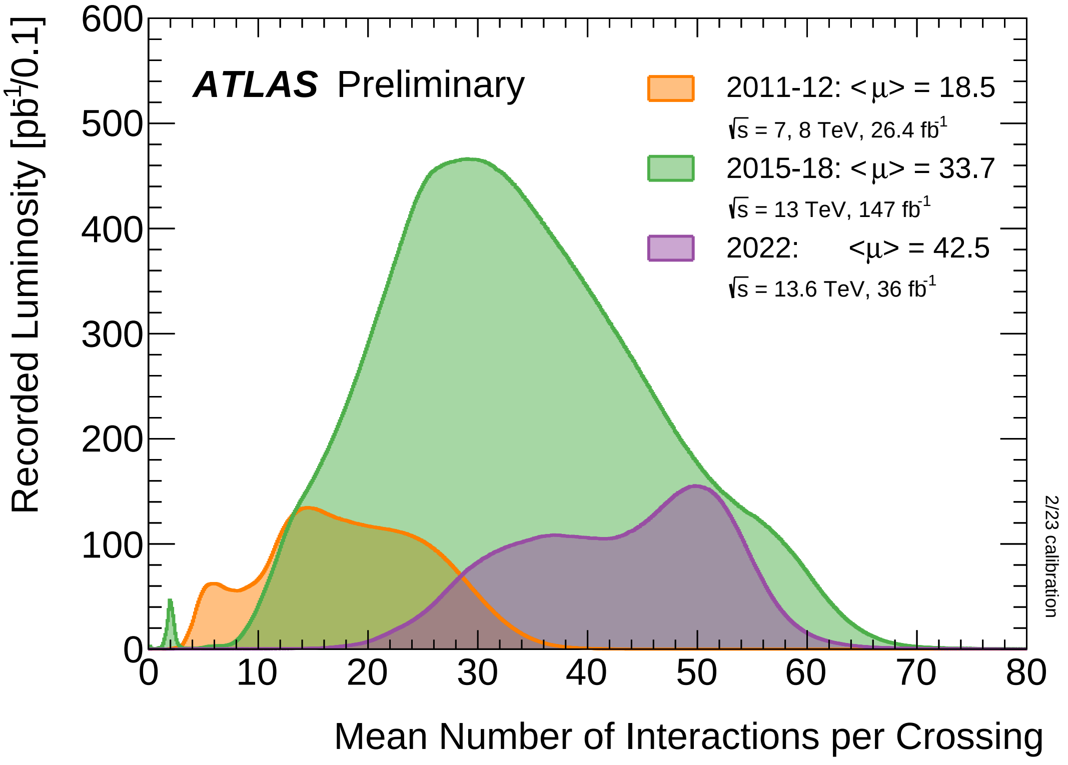


Figure 1.3: Shown is the luminosity-weighted distribution of the mean number of interactions per crossing for pp collisions at 7 and 8 TeV center-of-mass energy in 2011-12, 13 TeV in 2015-18, and 13.6 TeV in 2022 [11]. All physics data recorded by ATLAS during stable beams in these years are shown, and the integrated luminosity and the mean mu value are given in the figure. The mean number of interactions per crossing corresponds to the mean of the Poisson distribution of the number of interactions per crossing calculated for each colliding bunch pair. It is calculated from the instantaneous per bunch luminosity as $\mu = L_{bunch} \times \sigma_{inel}/f_r$ where L_{bunch} is the per bunch instantaneous luminosity, σ_{inel} is the inelastic cross section, and f_r is the LHC revolution frequency. The inelastic cross section has been assumed to be 71.5 mb at 7 TeV, 73 mb at 8 TeV, and 80 mb at both 13 TeV and 13.6 TeV. The luminosity shown for 2022 uses a preliminary 13.6 TeV luminosity calibration based on van-der-Meer beam-separation scans taken in 2022.

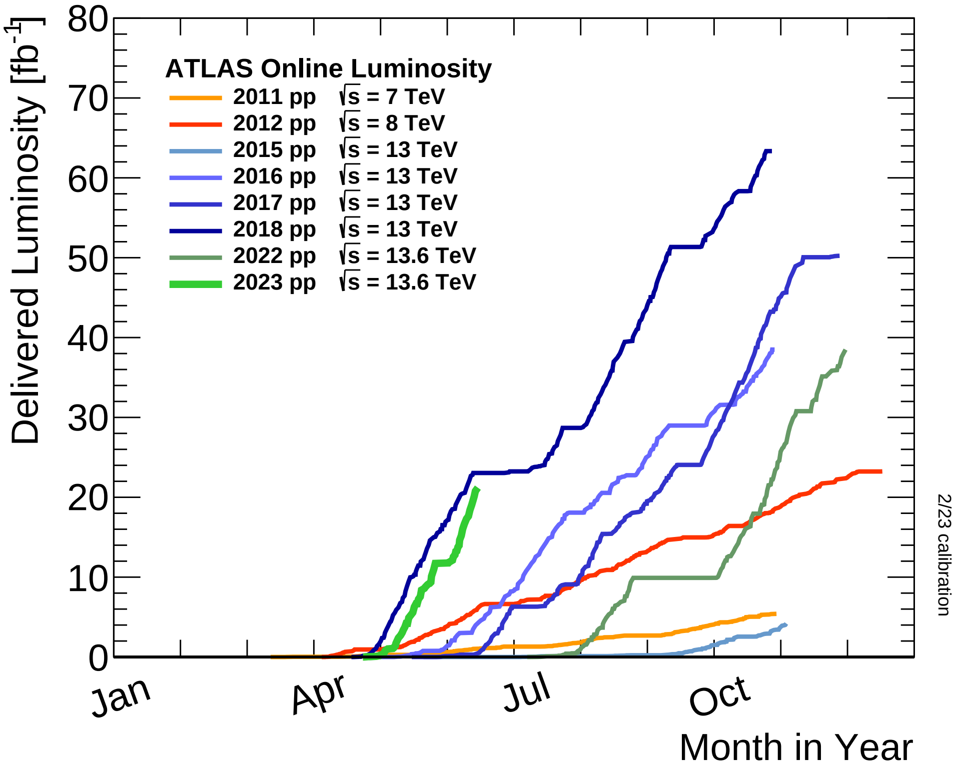


Figure 1.4: Cumulative luminosity versus day delivered to ATLAS during stable beams and for high energy p-p collisions [11].

1.2.3 High-luminosity LHC

While the LHC has been running since 2008, most of its data taking is still ahead. The LHC will undergo several upgrades during its lifetime as seen on figure 1.5. The last upgrade is the High Luminosity LHC (HL-LHC) upgrade [12]. To be able to fulfill its physics goals upgrades are required for precision measurements of the Higgs boson properties, other standard model processes and searches for phenomena beyond the standard model such as dark matter. The upgrade of the LHC will increase the luminosity while upgrades for the detectors will allow for more accurate data collection. Higher luminosity requires upgrading of numerous components in the LHC and its detectors to withstand higher radiation as well as upgraded electronics to track and store the increased amount of events. The upgrades will be installed during the long shutdown 3 (LS3) from 2026-2028 in what is known as the Phase II upgrade.

The target luminosity of the HL-LHC is $7.5 \cdot 10^{34}$ cm$^{-2}s^{-1}$ which corresponds to an average of 200 proton-proton collisions per bunch crossing ($\mu = 200$). An integrated luminosity of 300 fb^{-1} per year is foreseen. The scheduled duration of the HL-LHC is over 10 years and it is estimated that a total of 4000 fb^{-1} of data is produced, which is 10 times of what will be gathered during the combined runs 1-3.

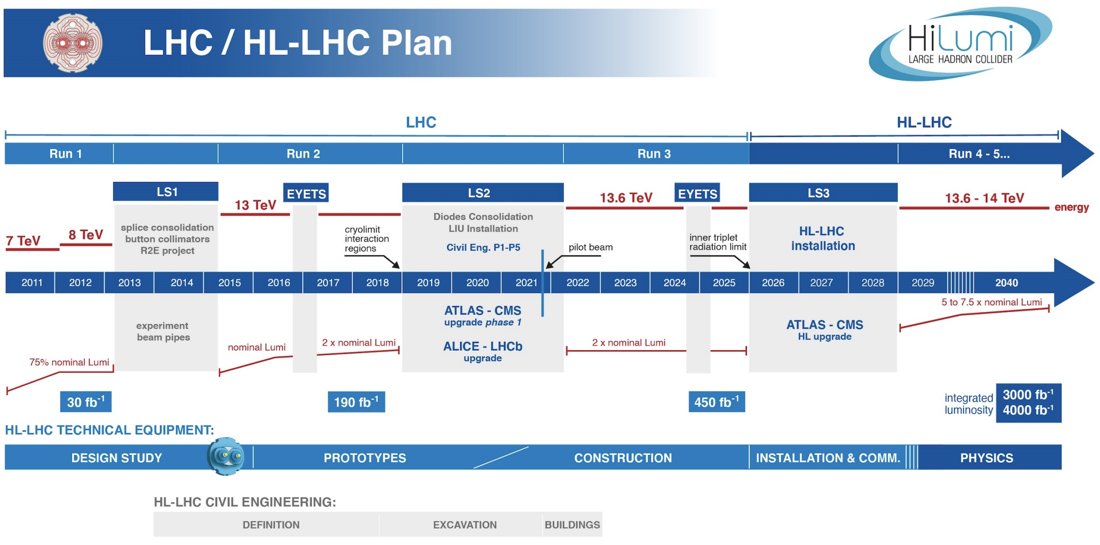


Figure 1.5: The timeline of upgrades at the LHC.

1.3 The ATLAS experiment

The ATLAS experiment [13] is one of the two large general-purpose experiments. It was designed to study proton-proton (pp) as well as heavy-ion collisions at the LHC. The detector is situated in an underground cavern near the CERN main site at point 1 seen in figure 1.2. The detector, shown in figure 1.6, has four main detector elements and the interactions of different particles with the ATLAS components are shown in figure 1.7. Each component is designed to detect specific particles or to measure the properties of specific particles.

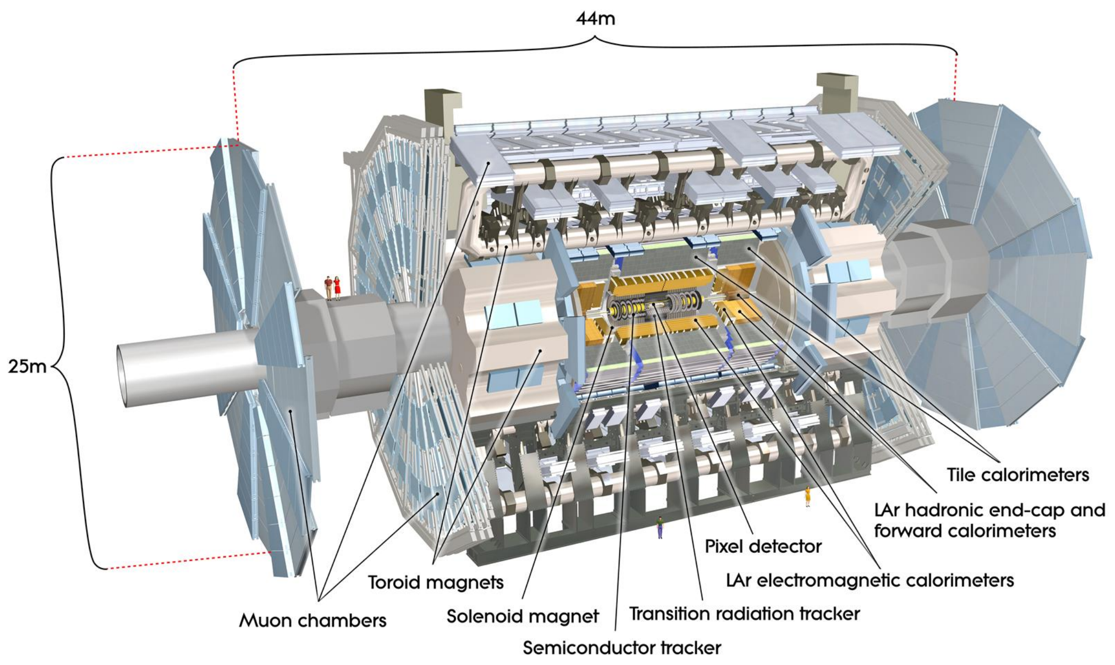

44m
25m
Tile calorimeters
LAr hadronic end-cap and forward calorimeters
Pixel detector
LAr electromagnetic calorimeters
Toroid magnets
Muon chambers
Solenoid magnet
Transition radiation tracker
Semiconductor tracker

Figure 1.6: A cutout of the ATLAS experiment.

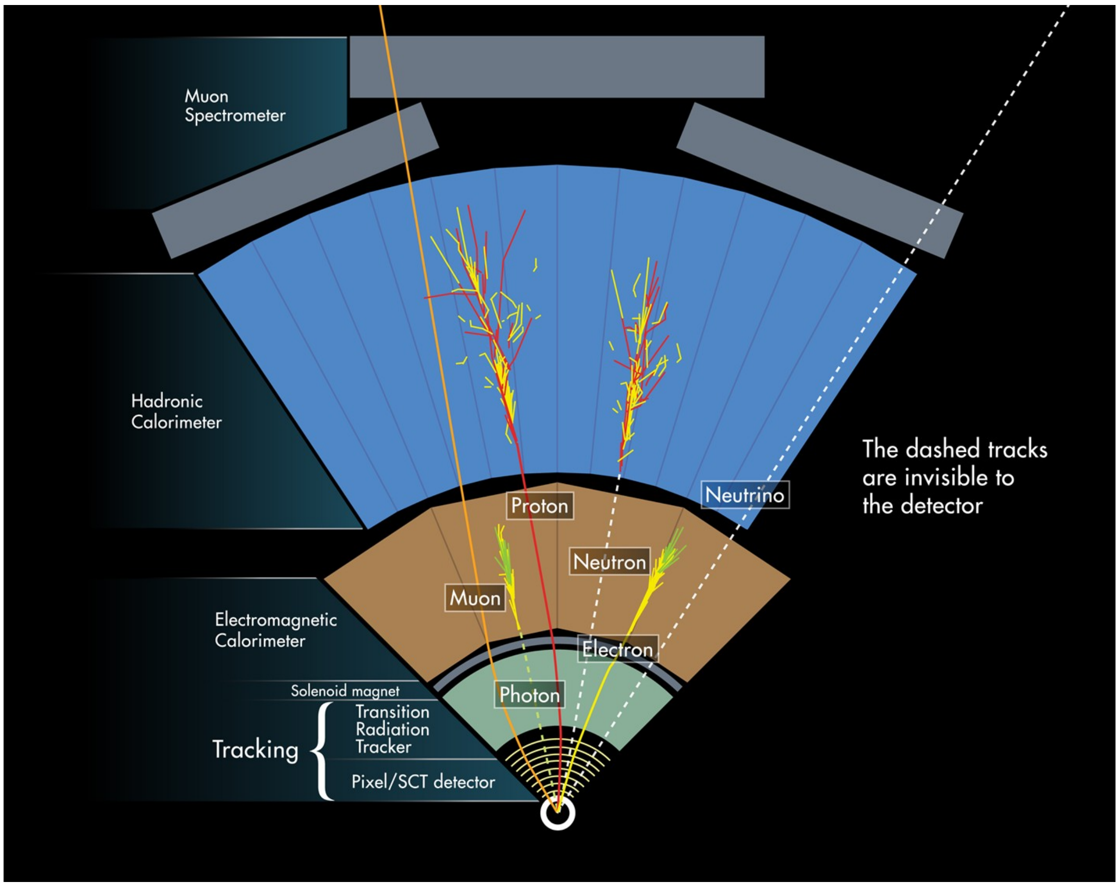


Figure 1.7: The interactions of different particles in the different sub-detectors of the ATLAS.

1.3.1 Coordinate System

ATLAS uses a right-handed coordinate system where the origin is the nominal interaction point with z-axis along the beam pipe, y-axis pointing upwards and x-axis points to the center of the LHC ring. The detector uses cylindrical coordinates which are defined as (θ, ϕ) where ϕ is the azimuthal angle around the z-axis and θ is the angle from the z-axis.

The commonly used variable instead of θ in hadron collider physics is the pseudorapidity η which is defined through the angle from the z-axis θ shown in equation 1.2. This is because the distribution of particles in pseudorapidity is more uniform than in the actual angle.

$$\eta = -ln(tan(\theta/2)) \tag{1.2}$$

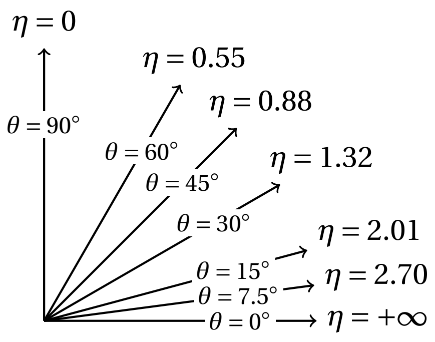


Figure 1.8: Pseudorapidity η for different θ angles [14].

Further important variables derived from η are the transverse energy defined as $E_T = E/cosh(\eta)$ as well as the angular distance defined as $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$.

1.3.2 Inner Detector

The ATLAS Inner Detector (ID) [15] is located closest to the interaction point and it measures the direction, momentum and charge of charged particles produced in the collisions as well as providing the primary and secondary vertex measurements [13]. The ATLAS inner detector consists of three different systems of sensors, shown in figure 1.9 that are immersed in a magnetic field that is parallel to the beam axis.

During the Phase-II upgrade of the inner detector [16], the Inner tracker (ITk) will be installed. The new detector replaces the existing ID and consists of a new all-silicon detector configuration, shown in figure 1.10, where pixel layers are located closest to the interaction point and they are complemented by layers of strip detectors that surround the pixel layers.

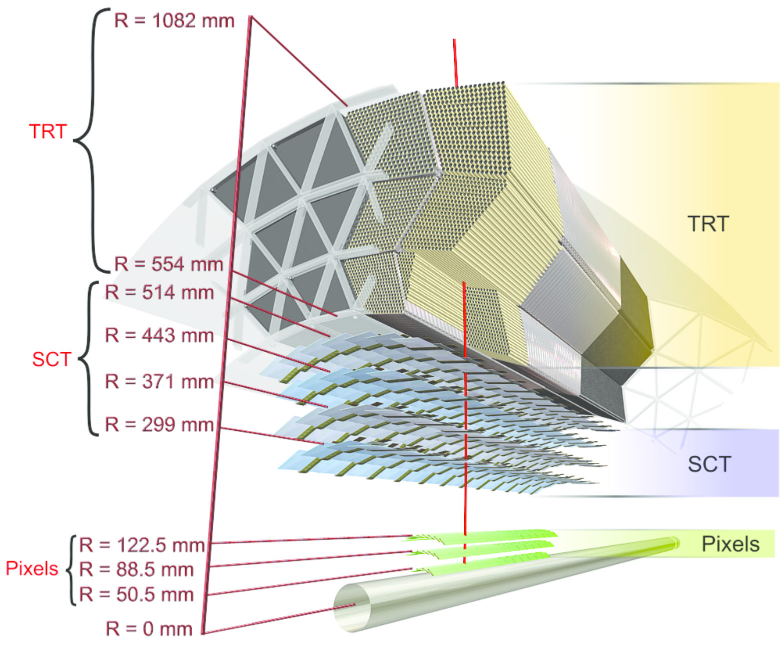


Figure 1.9: The figure shows the barrel part of ID of the ATLAS experiment with the Pixel, SCT and TRT sub-detectors [13].

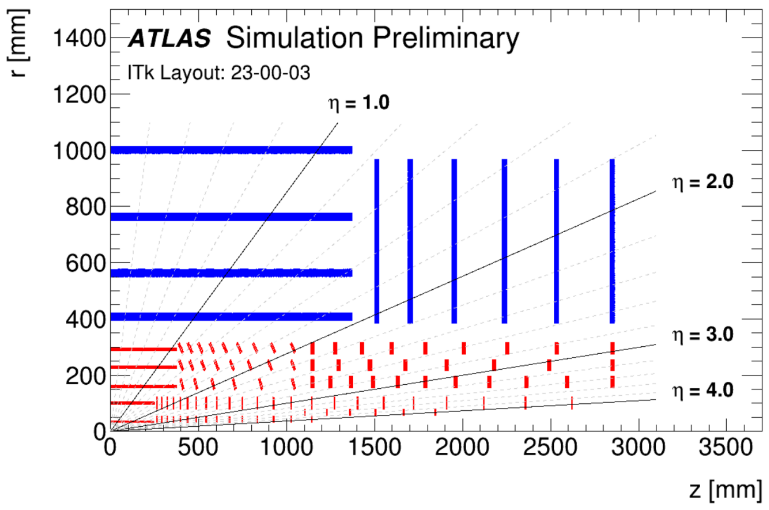


Figure 1.10: Schematic view of one quadrant of the ITk layout with only the active detector elements shown, in red for the pixel detector and in blue for the strip detector [17].

1.3.3 Liquid Argon Calorimeter

The liquid argon calorimeter (LAr) [18] is designed to trigger on and to provide precision measurements of photons, electrons, jets as well as missing E_T (MET). The calorimeter consists of several subsystems shown in the cutout figure 1.11. The calorimeter uses accordion shaped electrodes for all electromagnetic calorimetry up to $\eta < 3.2$, the construction is shown in figure 1.12. The construction forms a sampling calorimeter with absorbers made out of either lead-iron or copper-tungsten and liquid argon as the active material. The fraction of measured energy deposit in the active material of the total deposited energy is shown in equation 1.3. The energy is measured by the drift electrons ionized by passing particles attracted to the drift electrodes in the center of the accordion as shown in figure 1.13.

The different parts of the LAr are the Electromagnetic barrel (EMB) which covers $0 \leq |\eta| < 1.475$, Electromagnetic end-cap (EMEC) which covers $1.375 < |\eta| < 3.2$, Forward calorimeter (FCal) which covers $3.1 < |\eta| < 4.9$ as well as the Hadronic endcap (HEC) which complements the Tile calorimeter and measures hadrons in $1.5 < |\eta| < 3.2$. The absorbers of EMB and EMEC are made of lead-iron while the absorbers of HEC and FCal are copper-tungsten.

$$f_{sampling} = \frac{E_{active}}{E_{active} + E_{absorber}} \tag{1.3}$$

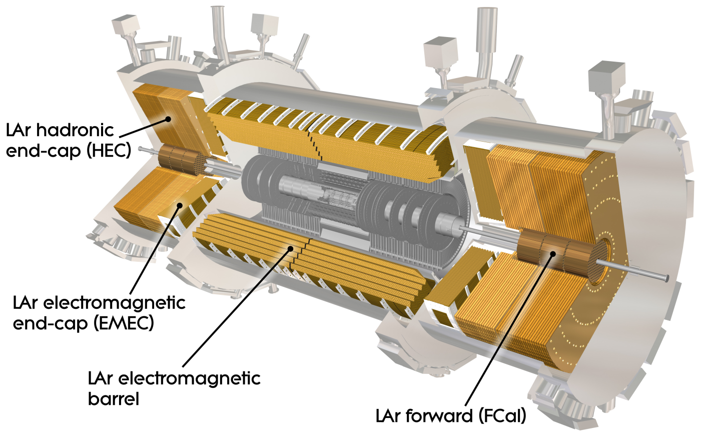

LAr hadronic end-cap (HEC)
LAr electromagnetic end-cap (EMEC)
LAr electromagnetic barrel
LAr forward (FCal)

Figure 1.11: A cutout of the LAr calorimeter with its subcomponents annotated [19].

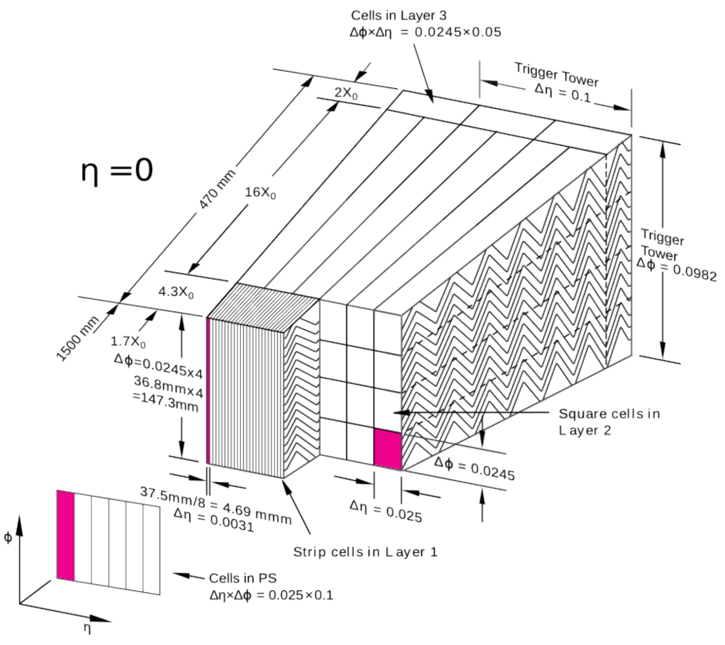


Figure 1.12: The layers of the LAr calorimeter module. The different modules are presampler: layer 0/PS, front: layer 1, middle: layer 2 and back: layer 3 [13].

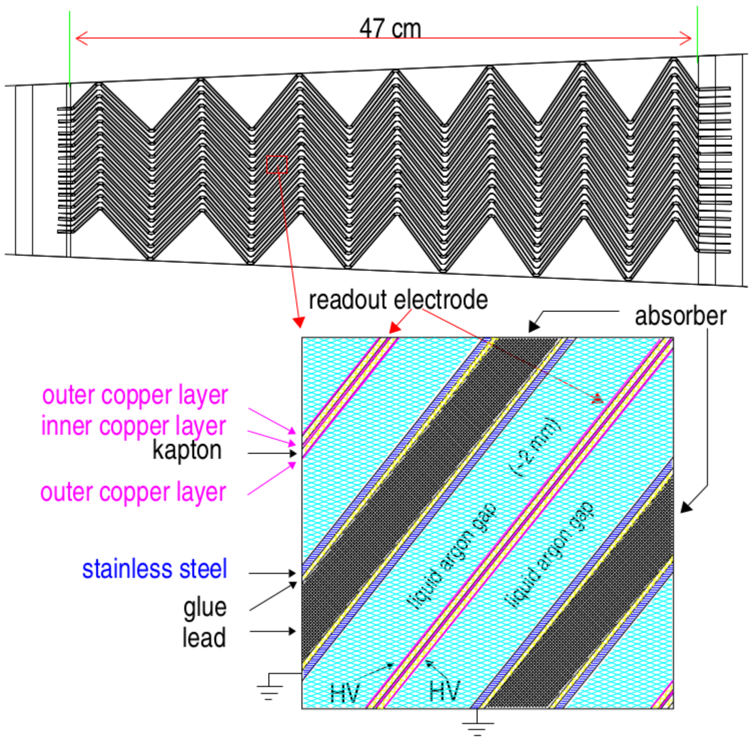


Figure 1.13: The LAr accordion structure with the active medium, absorber material and the readout electrode [18].

1.3.3.1 LAr calibration system

The LAr calorimeter requires constant calibrations for precision measurements which are done inbetween fills of the accelerator. The system is based on a pulsing board in the front-end crates. This pulsing creates a response similar to the response of ionization signal from particles depositing their energy in the calorimeter [20]. The known amplitude of the pulse is used for the calibration.

1.3.4 Trigger and Data-Acquisition System

The collision rate of the LHC is 40MHz [8]. This high event rate requires a multi-staged triggering system to record only the events with interesting physics, ultimately dropping the event data recording rate to an average of approximately 1 kHz. This system if referred to as the Trigger and Data Acquisition (TDAQ) system [21].

The trigger system undergoes several upgrades over its lifetime. For Run-2 it has two level structure: Level-1 and the high level trigger (HLT), each reducing the rate of data passed to the subsequent level [21]. The Level-1 trigger forms its decision by receiving information from the Level-1 calorimeter (L1Calo) and muon (L1Muon) triggers as well as from the topological (L1Topo) trigger. The Level-1 trigger reduces the data rate from 40MHz down to 100kHz and has a latency of $2.5\mu s$. Once the Level-1 triggers, a Level-1 Accept (L1A) signal is sent to the detectors, which will then send their buffered data to the data acquisition (DAQ) system. The HLT performs object reconstruction by using input from Level-1 in form of geometrical regions in the detector called regions of interest (RoI). It has an event accept or reject rate of 1 kHz and a latency of 200ms.

The HL-LHC upgrade will introduce changes to the trigger system. The Phase-I trigger system [22], commissioned in 2022, will run in parallel to the Phase-II triggers. The number of background events passing the threshold will increase as the number of interactions per bunch crossing goes up to $\mu = 200$ which requires improvements to the system. The main shortcomings are the low rate of 100 kHz which will not be sufficient for the higher pileup rates, the inability to process higher granularity information from the calorimeters as well as the muon trigger acceptance in the barrel region and the low background rejection of the muon trigger in the endcap regions. The Level-0 trigger and the Global Trigger seen in figure 1.14 are currently being developed [21]. The Level-0 hardware triggers will have a rate of 1 MHz and latency of $10\mu s$ using information from the calorimeters and the muon system.

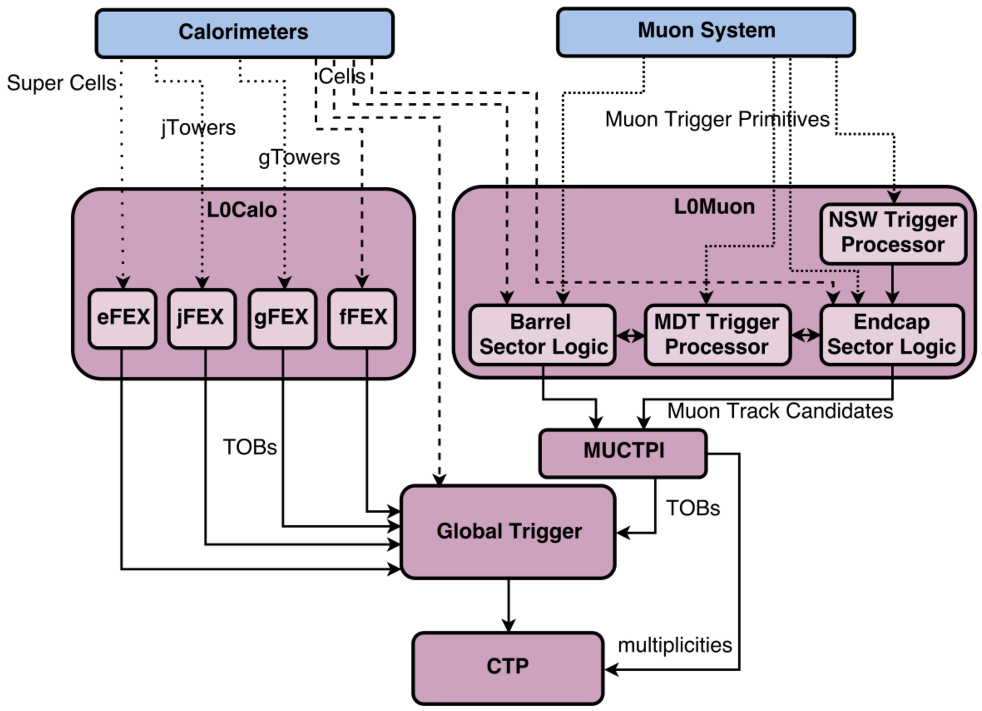


Figure 1.14: L0 trigger system supplying information to the central trigger processor (CTP).

1.3.5 Tile Calorimeter

The Tile calorimeter [23] is the hadronic calorimeter of the ATLAS experiment at $|\eta| < 1.7$ and measures the energy of hadrons as they pass through the detector and interact with matter. It is a sampling calorimeter using steel as the absorber and scintillator as the active medium. Ultraviolet scintillator light is induced by ionising particles crossing the medium, this is converted to visible light and measured using photomultiplier tubes (PMTs).

The main upgrades in Phase-II are that in the analogue electronics will be replaced and all data from the Tile cells will be sent to the trigger system at 40 MHz. In addition the old PMTs will be replaced with newer technology [24].

1.3.6 Muon spectrometer

The outer part of the ATLAS is the muon spectrometer [13]. It is designed to detect and to measure the momenta of charged particles at $|\eta| < 2.7$ and to trigger on the particles at $|\eta| < 2.4$. The different components shown in figure 1.15 are the Monitored Drift Tube chambers (MDTs) which are used for precision measurement of the momentum and in the forward region $2 < |\eta| < 2.7$. Cathode-Strip Chambers (CSC) are used in the inner-most tracking layer due to their capability for higher rates and better time resolution.

For the purpose of the triggering, track information is delivered within tens of microseconds after the passage of the particle. This is done using Resistive Plate Chambers (RPC) in the barrel region of $|\eta| < 1.05$ and Thin Gap Chambers (TGC) in the end-cap region $1.05 < |\eta| < 2.4$.

The main upgrade for Phase-II for the muon spectrometer is primarily focusing on the improvement of the muon trigger chambers [21]. The Level-0 trigger electronics of the RPC and TGC chambers will be upgraded. Also new RPC detectors will be added in the barred in the region of $|\eta| < 1$, which will significantly increase the solid angle coverage and the redundancy of the Level-0 trigger system. In addition the front-end readout of the MDT will be replaced to fulfill the trigger rate and latency requirements of the new trigger system as well as allow the hit information from MDT to be used in the Level-0 trigger which improves the muon p_T resolution in the trigger.

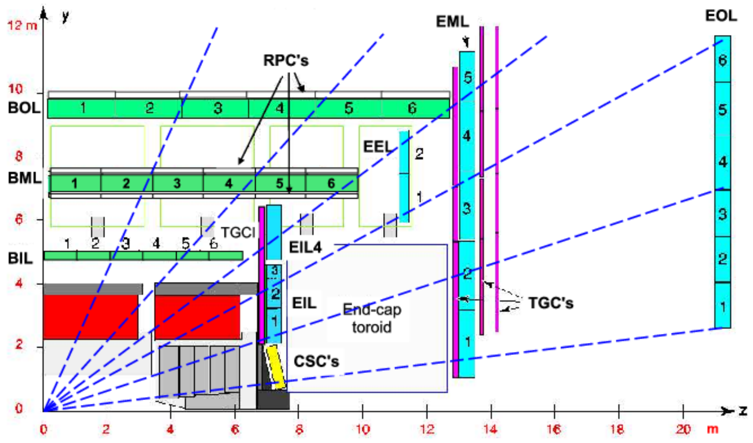


Figure 1.15: Cross-sectional view of ATLAS muon spectrometer. The barrel MDT chambers are shown in green, the endcap MDT chambers are blue. In the barrel (endcaps), the RPC (TGC) chambers are shown outlined in black (solid purple) [13].

1.4 LAr Phase-II Upgrade

The main objectives for the LAr Phase-II upgrade [25] are to replace the LAr readout electronics. The new readout electronics are required because of the trigger upgrade, which operates at a frequency of 1MHz. In addition the trigger granularity is increased to cell level granularity. Much of the current readout system has been in use since run-1 and consists of obsolete electronics and furthermore components inside the detector have been degraded by radiation.

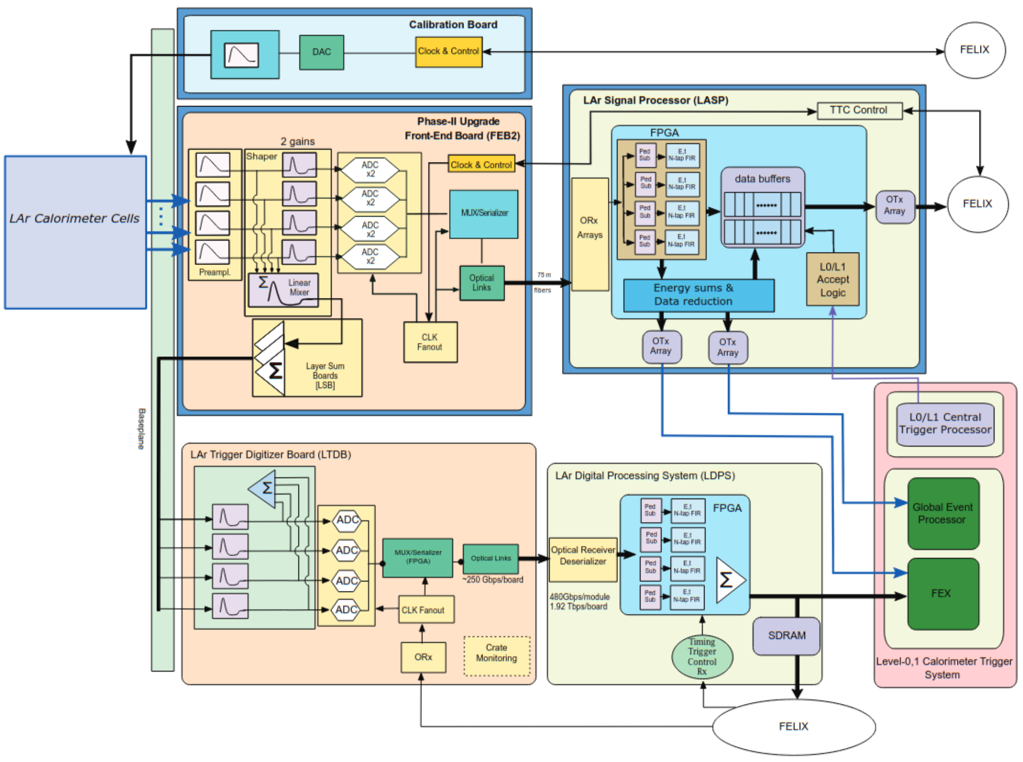


Figure 1.16: Schematic of the LAr calorimeter readout architecture for the Phase-II upgrade [25].

1.4.1 Front-end Electronics

The front-end electronics are radiation hardened components located inside the detector, mounted on the LAr cryostats. The electronics are housed in front-end crates (FECs) which will host the upgraded front-end boards (FEB2s). These boards receive the input of 128 LAr channels, shape the incoming analog signal with $CR-(RC)^2$ filter, digitize it and this signal is sent to the computing room using optical links. The board will host custom application specific integrated circuits (ASICs). This includes ASICs for the analog filtering and for the analog-to-digital (ADC) conversion. In addition new calibration boards will be installed in the FECs.

1.4.2 LASP

The LAr Signal Processor (LASP) is the board that receives the digitized signal from the FEB2s. The board will reconstruct the energy and the phase of the pulses using digital filtering techniques, supply the information to the trigger while buffering the data waiting for the triggering decision and if an accept is received the board will then transmit the data to the Global Event Trigger Processors and the L0Calo systems.

Each LASP board will process 384 LAr channels. The board is required to compute the energies for all of the channels within 125 ns as well as to buffer the energies in case of the accept from the trigger for up to 10 μs. The specifications also require the system to be capable of bursts of 4 L0 accepts in 5 consecutive BCs, 8 L0 accepts in 5 μs and 128 L0 accepts in 30 μs. The high throughput computing of LASP is done with an Intel AgileX FPGA. In this work energy reconstruction algorithms based on neural networks are evaluated to be deployed on the LASP in place of the analytical method used up to now.

1.4.3 Legacy energy reconstruction algorithms

In the liquid argon calorimeter, particles passing the detector ionize the liquid argon that acts as the active material. The free electrons drift by high voltage and are collected by electrodes [26]. The detector response is a triangular detector pulse shape shown in figure 1.17. This response has a fast rise and tail that lasts for up to 500 ns. It is a highly stable response that varies only at its peak height which is proportional to the deposited energy and the peaking time. These two factors, the height and time of the peak, are what the reconstruction in LAr aims to compute. With a triangular pulse and sampling rate of 40MHz, the precise peak height and timing will be missed and cannot be inferred from the sampled signal to counter this an analog $CR-(RC)^2$ filter is used to filter the signal into a known shape which allows for accurate reconstruction of the height and timing. The response after applying the filter is a bipolar pulse which is sampled at 40 MHz is seen in figure 1.17.

1.4.3.1 Optimal Filtering

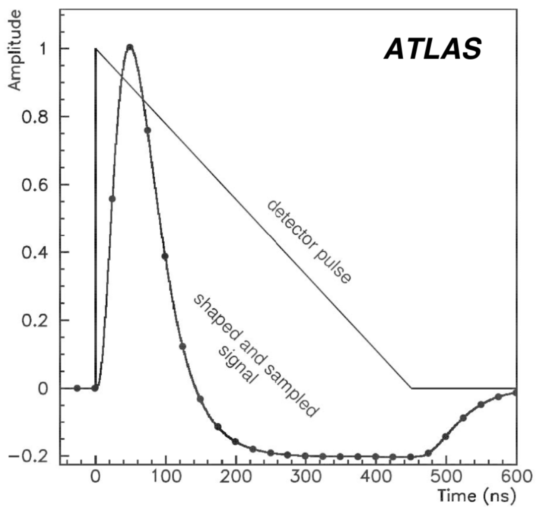


Figure 1.17: LAr pulse shape using a $CR-(RC)^2$ sampled at 40 MHz.

The Optimal Filtering (OF) [27] algorithm is used in the LAr calorimeter to extract the amplitude and timing from a noisy signal. This algorithm allows to minimize the effects of pileup and other sources of noise such as thermal noise.

The amplitude of the pulse A is reconstructed with formula 1.4 and the timing with 1.5 as the weighted sum of the digitized signal samples and filter coefficients. Filter coefficients a_i are used for the pulse amplitude and coefficients b_i are used for the timing τ which measures the phase shift between the expected and actual pulse timing. The reconstruction is done with either $n = 4$ or $n = 5$ samples depending on the run. The studies in this work are done with $n = 5$.

$$A = \sum_{i=1}^{n} a_i \cdot s_i \tag{1.4}$$

$$A\tau = \sum_{i=1}^{n} b_i \cdot s_i \tag{1.5}$$

1.4.3.2 Continuous energy measurement with OF

The duration of the pulse spans across several bunch crossings. A maximum finder is used on the optimal filtering output to assign the energy E to a specific bunch crossing [28]. The condition for attributing the energy to a bunch crossing is that it is the peak energy of the preceeding as well as following bunch crossing and above a selected threshold E_{thr}. Otherwise a zero energy is attributed for that bunch crossing. The maximum finder condition is formulated in equation 1.6 where the energy is assigned to a BC i if E_i is larger than E_{i-1} and E_{i+1} [29]. The combination of optimal filtering with maximum finder is referred to as OFMax. The increased luminosity of HL-LHC will introduce pile-up conditions that significantly reduce the performance of the Optimal Filtering algorithm and require new signal processing techniques [26].

$$E = \begin{cases} E_i & \text{if} \quad (E_i > E_{i+1}) \wedge (E_i > E_{i-1}) \wedge (E_i > E_{thr}) \\ 0 & \text{otherwise} \end{cases} \tag{1.6}$$

1.4.3.3 Other filter types for energy reconstruction

Optimal filtering struggles in HL-LHC conditions [29]. When using the OF, the peak height of a pulse can be biased by the trailing undershoot of a preceding pulse, leading to an underestimation of the preceding peak value of the pulse by up to 20% (depending on the pulse shape). In severe cases, the undershoot can completely obscure an energy deposit. To overcome the performance drop other filter types have been evaluated in previous works [29]. The best performing filter type considered as an alternative to the OFMax is the Wiener Filter with Forward Correction (WFFC) which has a specific focus on effectively detecting overlapping signal contributions. The goal of the WFFC is to minimize this effect while retaining the ability to accurately reconstruct the energy deposits for each BC. While the WFFC and OFMax perform similarly overall, the WFFC is able to reconstruct energies better in the case of overlapping signals while OFMax performs better in the isolated scenario.

The evaluation in this work is only done in comparison to OFMax because the goal is to improve the performance in both the case of isolated pulses as well as overlapping ones.

1.5 ATLAS Simulation

ATLAS simulations can be done either by simulating the LAr readout electronics using the ATLAS readout simulator (AREUS) [30] or using the full detector simulator Athena [31]. AREUS simulation produces a continuous signal for a single calorimeter cell whereas Athena produces the detector response for selected physics events for the full detector without continuity between the events.

1.5.1 AREUS

AREUS is a simulation framework of the continuous signal of the LAr readout electronics for individual LAr cells, which allows the evaluation of various digital filtering algorithms. The simulation is done BC-by-BC basis over a period of time and the detector response is simulated as it would happen in the continuous data flow in the detector. This continuous nature enables the filter algorithms to show their potential unexpected behavior in long propagation time as it would be the case in data-taking, which includes ringing, oscillations, swinging up and baseline shift.

1.5.2 Athena

In Athena events are generated and processed on an event-by-event basis. The general digital processing is based on the time grid defined by bunch crossings (BCs), but the processing of events is done independently of each other: for each event it is possible to use in a total of 32 BCs around the event by ATHENA when calculating its filter result. Intermediate filtering results are then discarded and the processing is recycled for the next event which is again computed independently from the previous event. This processing method is excellent when evaluating the detector performance with selected physics events, but has limitations when evaluating filter algorithms.

1.6 Electron and photon reconstruction and identification

Out of the deposited energy and tracks in the detector it is possible to reconstruct the energy and identify the particles in question as shown in figure 1.7. The reconstruction algorithm introduced in 2017 is based on variable-sized clusters referred to as superclusters [32]. This algorithm replaces the sliding-window algorithm [33]. The main improvements are dynamic clusters that change in size as needed to recover energy from bremsstrahlung photons or from electrons from photon conversions. An important reason for using superclusters is the improved energy resolution that superclusters provide by collecting more of the deposited energy. The reconstruction of electrons and photons with $|\eta| < 2.5$ proceeds as shown in figure 1.18.

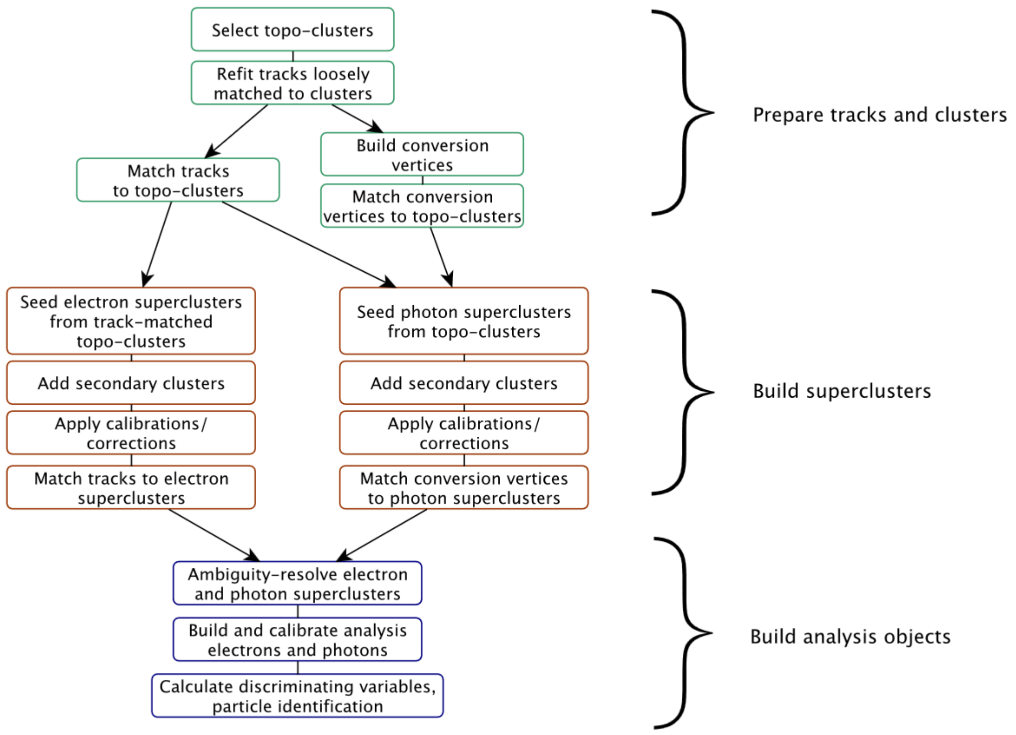

Select topo−clusters
Refit tracks loosely matched to clusters
Match tracks to topo−clusters
Build conversion vertices
Match conversion vertices to topo−clusters
Prepare tracks and clusters
Seed electron superclusters from track−matched topo−clusters
Add secondary clusters
Apply calibrations/ corrections
Match tracks to electron superclusters
Seed photon superclusters from topo−clusters
Add secondary clusters
Apply calibrations/ corrections
Match conversion vertices to photon superclusters
Build superclusters
Ambiguity−resolve electron and photon superclusters
Build and calibrate analysis electrons and photons
Calculate discriminating variables, particle identification
Build analysis objects

Figure 1.18: Algorithm flow diagram for the electron and photon reconstruction. [32]

1.6.1 Topo-cluster reconstruction

The first part of the algorithm is to prepare tracks and clusters to be used. It first builds EM topo-clusters by forming proto-clusters in the calorimeters using the cells that have deposits above a set of noise thresholds. The initial step uses a threshold of $E^{EM}_{cell}/\sigma^{EM}_{noise,cell} \geq 4$.

In the next iteration proto-clusters then collect neighboring cells with half the significance ($E^{EM}_{cell}/\sigma^{EM}_{noise,cell} \geq 2$) and these become seed cells for the next iteration. If two proto-clusters contain the same cell, the clusters are merged. In the final step a crown of nearest-neigbour cells are added to the cluster without applying thresholds. This is known as '4-2-0' set of thresholds. The proto-cluster is then split if there are local maxima that fulfill the conditions: $E^{EM}_{cell} > 500 MeV$ with at least four neighbors none of which have a higher signal.

A preselection is applied to cut up to 60% of pile-up clusters without affecting the true electron topo-clusters. This preselection is based on cuts on the EM energy of the cluster, which is measured with the cluster energy using cells of the EM calorimeter (in the transition region also scintillator and presampler are taken into account) larger than 400 MeV and the fraction of EM energy to the total energy of the cluster $f_{EM} > 0.5$.

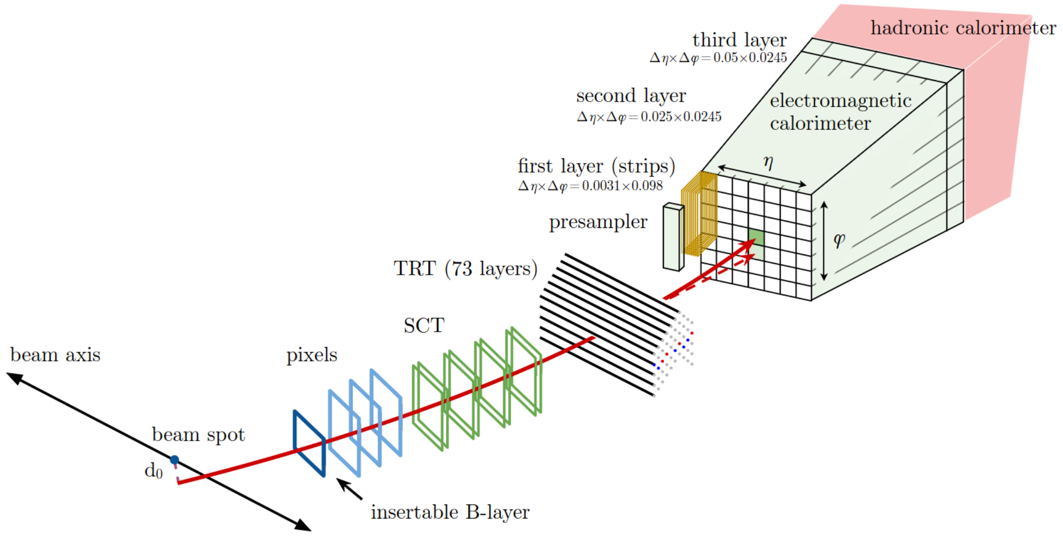

third layer
$\Delta\eta\times\Delta\varphi = 0.05\times0.0245$
hadronic calorimeter
second layer
$\Delta\eta\times\Delta\varphi = 0.025\times0.0245$
electromagnetic calorimeter
first layer (strips)
$\Delta\eta\times\Delta\varphi = 0.0031\times0.098$
η
presampler
φ
TRT (73 layers)
SCT
beam axis
pixels
beam spot
d_0
insertable B-layer

Figure 1.19: A schematic illustration of the path of an electron through the detector. The red trajectory shows the hypothetical path of an electron, which first traverses the tracking system (pixel detectors, then silicon-strip detectors and lastly the TRT) and then enters the electromagnetic calorimeter. The dashed red trajectory indicates the path of a photon produced by the interaction of the electron with the material in the tracking system [34].

1.6.2 Track reconstruction

Track reconstruction is done in the inner detector as described in [34] with improvements introduced in [32]. The tracks are reconstructed from charged particle hits in the inner detector as shown in figure 1.19. The tracks are reconstructed using pattern reconstruction [35], χ^2 fitter [36] or Gaussian sum filter (GSF) [37] which is a non-linear generalization of the Kalman filter [38]. The algorithm used depends on properties and success of other methods. The acquired tracks are fitted and matched with the EM topo-clusters. In case multiple tracks are matched to a cluster, they are ranked. Highest rank is given to tracks that have hits in the pixel detector, then tracks that have hits in the SCT but not in the pixel detector. Inside these categories tracks are ranked by better matching ΔR to the cluster in the second layer of the calorimeter. The track with highest ranking is used to define the properties of the reconstructed electron.

1.6.3 Supercluster reconstruction

Superclustering gathers adjacent EM topo-clusters that originate from the same source. The satellite clusters may emerge from bremsstrahlung radiation or topo-cluster splitting. The superclustering is done independently for electrons and photons in two stages. Figure 1.20 shows the process of the superclustering where electron supercluster seed requires EM energy $E_T \geq 1GeV$ and must be matched to a track with at least four hits in the silicon tracking detectors. Photon seeds require larger EM energy of $E_T \geq 1.5GeV$ with no track requirements. The clusters are initially sorted by descending energy and the same cluster cannot be used as a seed if it has already been assigned as a satellite of another cluster.

For photons, a cluster is considered a satellite if it falls within a window of $\Delta\eta \times \Delta\phi = 0.075 \times 0.125$ around the seed cluster barycenter. For electrons the window is $\Delta\eta \times \Delta\phi = 0.125 \times 0.300$ and the best matched track is used for the supercluster.

For photons with conversion vertices made up only of tracks containing silicon hits, a cluster is added as a satellite if its best-matched (electron) track belongs to the conversion vertex matched to the seed cluster. Finally calorimeter cells are assigned for the supercluster using LAr layers 0-3 (for transition region $1.4 < |\eta| 1.6$ also scintillators are used).

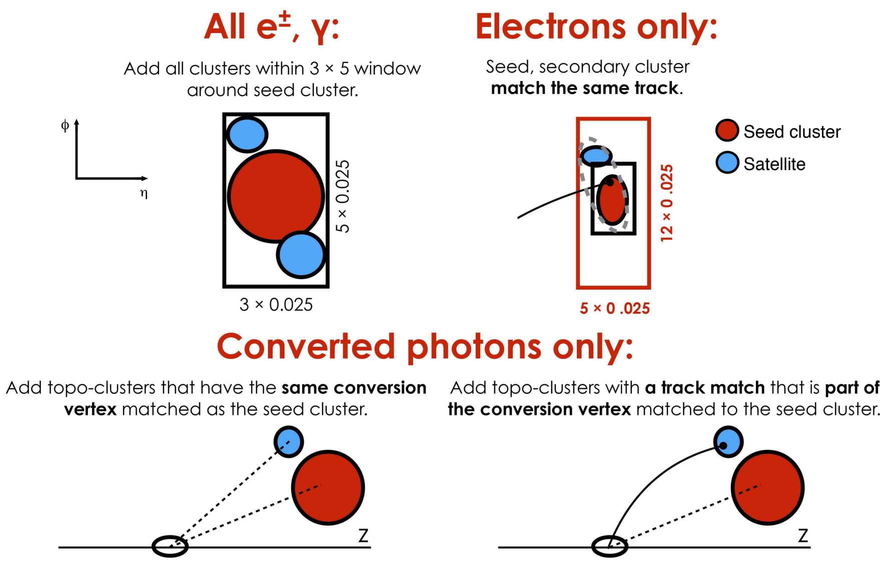

All e±, γ:
Add all clusters within 3 × 5 window around seed cluster.
ϕ
η
5 × 0.025
3 × 0.025
Electrons only:
Seed, secondary cluster match the same track.
12 × 0 .025
5 × 0 .025
Seed cluster
Satellite
Converted photons only:
Add topo-clusters that have the same conversion vertex matched as the seed cluster.
Add topo-clusters with a track match that is part of the conversion vertex matched to the seed cluster.
Z
Z

Figure 1.20: Diagram of the superclustering algorithm for electrons and photons. Seed clusters are shown in red, satellite clusters in blue [32].

1.6.4 Creation of analysis objects and performance

Finally analysis objects need to be created. An initial calibration is performed on the created superclusters and track matching is done for the electron superclusters. However as the electron and photon superclusters are built independently, a given seed cluster can produce both an electron and a photon. In the case of duplicate, an algorithm identifies the object. The object is classified for analysis as only a photon if there is a cluster but no good tracks or only an electron if there is a cluster with a good track and no good photon conversion vertex. Otherwise both electron and photon analysis objects are created. In addition these cases where the identification is ambiguous the objects are marked so for specific analysis requirements. The created objects are recalibrated and the shower shapes and other discriminating variables are computed.

Figure 1.21 shows the efficiency of the reconstruction for electrons as a function of E_T. The achieved efficiency is good and approaches the track reconstruction efficiency at higher p_T, but deteriorates at lower energies due to tracks which are matched to superclusters only GSF re-fit has been performed on them, this introduces a E_T threshold because of the fixed-size clusters. The performance of the reconstruction is affected by pileup levels. Figure 1.22 shows the electron identification efficiency as a function of pile-up. The efficiency decreases with higher pileup.

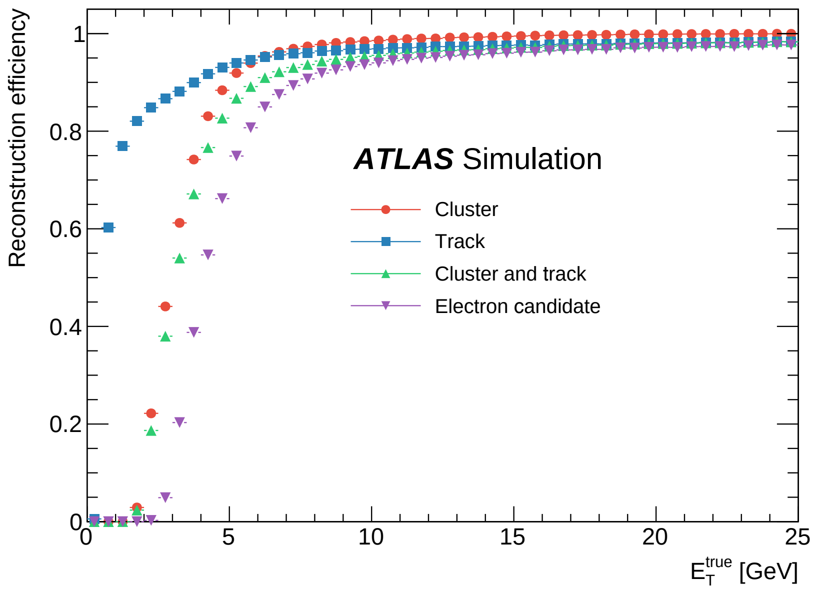


Figure 1.21: The cluster, track, cluster and track, and electron reconstruction efficiencies as a function of the generated electron E_T.

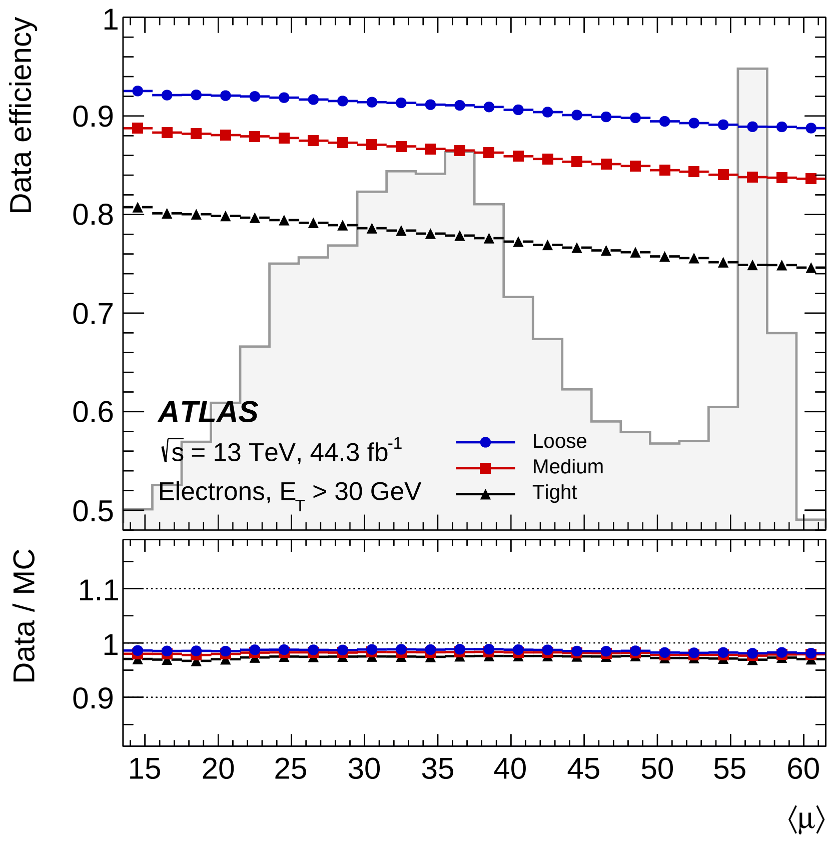


Figure 1.22: The electron identification efficiency in data for electrons with ET > 30 GeV as a function of the average number of interactions per bunch crossing for the Loose, Medium and Tight operating points. The efficiencies are measured in $Z \rightarrow ee$ events in data recorded in the year 2017. The shape of the η distribution is shown as a shaded histogram. The bottom panel shows the data-to-simulation ratios. The total uncertainties are shown.

2 Neural Networks

Summary

2.1 Introduction

Machine learning is a subset of artificial intelligence where a task is solved by using a learning algorithm that observes data and learns from it [39]. The system learns to find statistical structures of data in a process referred to as training where the input is transformed into meaningful output by using the observed error in the task to adjust how the task is solved.

The recent popularity of machine learning methods can be attributed to the availability of large datasets combined with new computing hardware. While machine learning is related to mathematical statistics, there are key differences. A key difference is the ability to deal with larger and more complex datasets, for example datasets that consist of millions of images each of which contain tens of thousands of pixels. The ability to learn meaningful representations from complex data can be attributed to the methods categorized as deep learning, where the data processing has a layered structure with a significant depth. These methods include recurrent neural networks (RNNs), convolutional neural networks (CNNs) and more recently the transformer architecture. The field of machine learning is also more engineering-oriented where ideas are often proven empirically rather than theoretically and often has little mathematical theory in comparison to classical statistical analysis.

The three main categories of machine learning problems are: supervised learning, unsupervised learning and reinforcement learning [40]. Supervised learning learns patterns from labeled data where each training sample has a known true value. Supervised learning tasks are divided to classification, where the output is a discrete class, and regression tasks where the output is real-valued. Unsupervised learning deals with unlabeled data from which patterns are learned, examples of unsupervised learning are clustering and dimensionality reduction algorithms. Reinforcement learning deals with an agent that takes actions in an environment and receives feedback of the performed action to learn. The learning in all of these categories requires a way to evaluate how well a task was performed, this is done using a loss function.

Machine learning techniques have been successfully used in high-energy physics experiments for different tasks such as particle identification, event selection, jet tagging or reconstruction at later stages of data analysis using methods such as Boosted Decision Trees, RNNs and CNNs [41]. However the ability to use advanced machine learning techniques in real time at the data acquisition and trigger levels has not been possible due to the restrictions imposed by the available hardware.

2.2 Neural Networks

2.2.1 Perceptron

The perceptron [42] is the earliest development of a biologically inspired structure for computation. It maps directly multiple weighted inputs into a binary value using a threshold function. The original Mark I perceptron was a physical machine built in 1958 and it was used for image recognition. The weights were adjusted using potentiometers. The weight adjustments were a heuristic process where weights were adjusted based on the error. However the original perceptron does not include a loss function. The training process has similarities to gradient descent, but is not derived from it as the perceptron is not a smooth function where a gradient-based method would work. While the perceptron was revolutionary for its time and laid the foundation for the field of artificial neural networks, it had serious limitations that made it applicable only for simple problems. The main limitation is that it is only able to converge with linearly separable data.

The mathematical representation of the perceptron is shown in equation 2.1. The sum of the inputs x_i are multiplied by weights w_i, the bias b is added and this is passed to the threshold function $thrs$. The schematic of the perceptron computation is shown in figure 2.1.

$$\hat{y} = thrs(\sum_{i=1}^{n}(w_i x_i) + b) \tag{2.1}$$

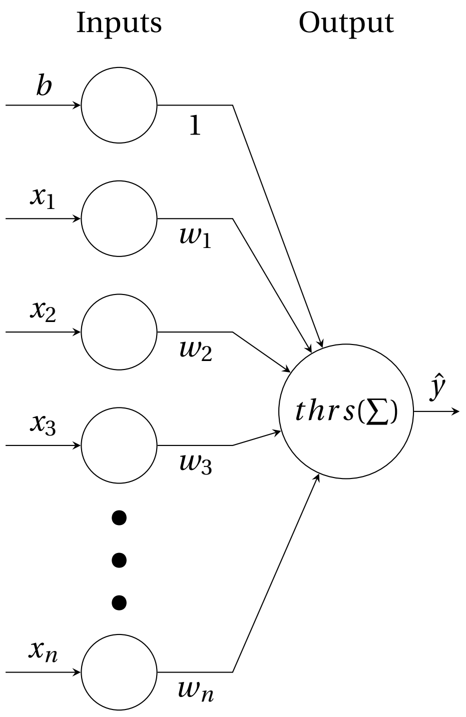


Figure 2.1: Computation of the perceptron. The inputs x_n are multiplied with the weights w_n and bias b and summed, the sum is passed through threshold function $thrs()$ to acquire output $\hat{y}$.

2.2.2 Feed Forward Network

The feed forward network, also known as the multi layer perceptron or dense networks, consists of neuronal units, commonly referred to as just units, similar to the perceptron organized in layers. Each unit has a weighted connection to every unit of the previous layer, thus the amount of weights in a given layer is $n \times m$ where n is the unit count of the previous layer and m the unit count of the layer itself. Each unit has also a bias b as well as a non-linear activation function. Equation 2.2 shows the computation of a single unit of a feed forward network where the weighted sum of inputs x_i multiplied by weights w_i is added to bias b and the result is passed through activation function $activation$. Figure 2.2 shows how the neurons are used to construct a feed forward network that has three layers.

$$\hat{y} = activation(\sum_{i=1}^{n}(w_i x_i) + b) \tag{2.2}$$

The addition of a non-linear activation function for each layer enables the network to approximate complex functions. Without a non-linear activation a feed forward network could be represented with a single linear layer and thus would not provide better modeling ability than a single layer linear network. This type of network is the fundamental building block of modern deep learning models due to the fact that other network types are either specialized usage of feed forward neurons such as the CNN, extensions of feed forward layers with feedback loops such as the RNN or are partly built from feed forward layers such as the transformer architecture.

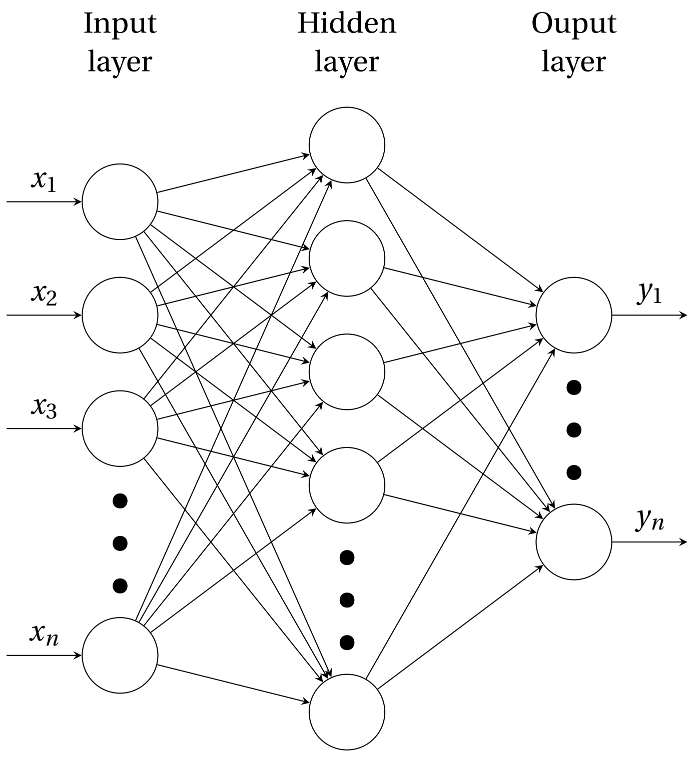


Figure 2.2: A schematic illustration of a feed forward network with a single hidden layer.

2.2.2.1 Activation functions

Activation functions are what enables neural networks to learn non-linear relations in the data. The key features of an activation function is that it is continuous and differentiable. Common activation functions are the sigmoid, hyperbolic tangent (tanh), rectified linear unit (ReLU) [43] and softmax. The mathematical formula and ranges of these activation functions are shown in figure 2.1 and their visualizations in figure 2.3 with the exception of softmax, because its value is dependent on the other outputs of the layer and thus cannot be plotted individually. The activation function chosen for the last layer of a network has special considerations due to its effect on the range and other properties of the output of the network.

Sigmoid was more commonly used in the past but is nowadays mostly used as the final activation in binary and multi-label classification, this is due to its range of $x \in [0,1]$ which can represent a probability. It is also used in RNNs.

Activation	Formula	Range
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$(0,1)$
Tanh	$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$(-1,1)$
ReLU	$f(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{otherwise} \end{cases}$	$[0,\infty)$
Softmax	$\Phi(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)}$	$(0,1)$

Table 2.1: Common activation functions, their mathematical expressions and ranges.

Tanh was a commonly used activation function but is nowadays mainly used in the output layer of some generative models due to its output range $x \in (-1,1)$. It is also used in RNNs.

ReLU has become one of the most popular activation functions due to its simplicity. The downside of the activation function is that it might "die" during the training or due to a bad neuron weight initialization or adjustment because the gradient of the negative part is zero. If this happens the learning of the neuron stops. Variations include the leaky ReLU [44] with non-zero negative part and the Gaussian Error Linear Unit (GELU) [45].

Softmax is the activation function used in the last layer of multi-class classification. It turns the output values into a probability distribution that sums to 1. It is also used in the transformer architecture as an intermediate activation function.

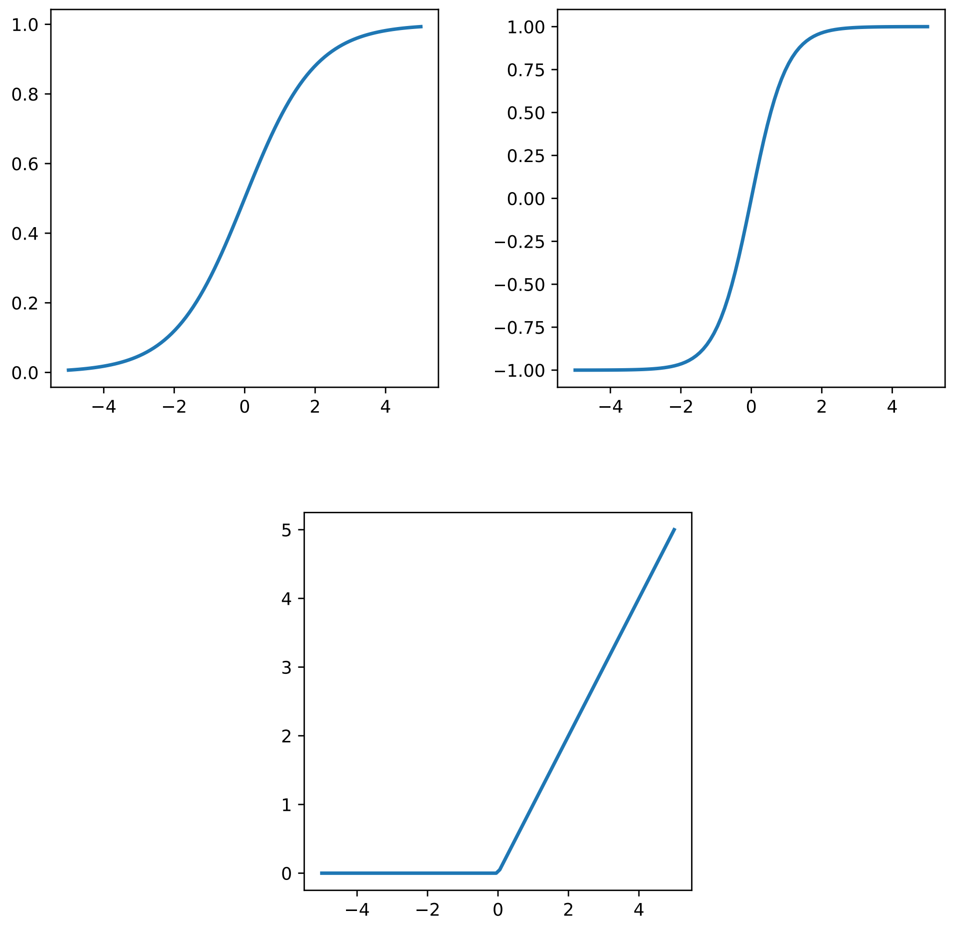


Figure 2.3: Common activation functions. (top left) sigmoid, (top right) hyperbolic tangent (tanh), (bottom) rectified linear unit.

2.2.3 Optimization

The training of a neural network consists of minimizing the loss function by adjusting the weights and biases. This is done by an optimizer. The weights of a neural network are initialized with random values while biases are often initialized to zeros. The predictions will at first be random, but during the training process the optimizer adjusts the weights and biases to minimize the error between the target and predicted values which is quantified by the loss. The training loop, illustrated in figure 2.4 shows how the predicted values along with the true values are used as inputs of the loss. The loss value is then used in the optimizer to perform weight adjustments. The training is done in batches and the loss value is averaged over a batch of training samples.

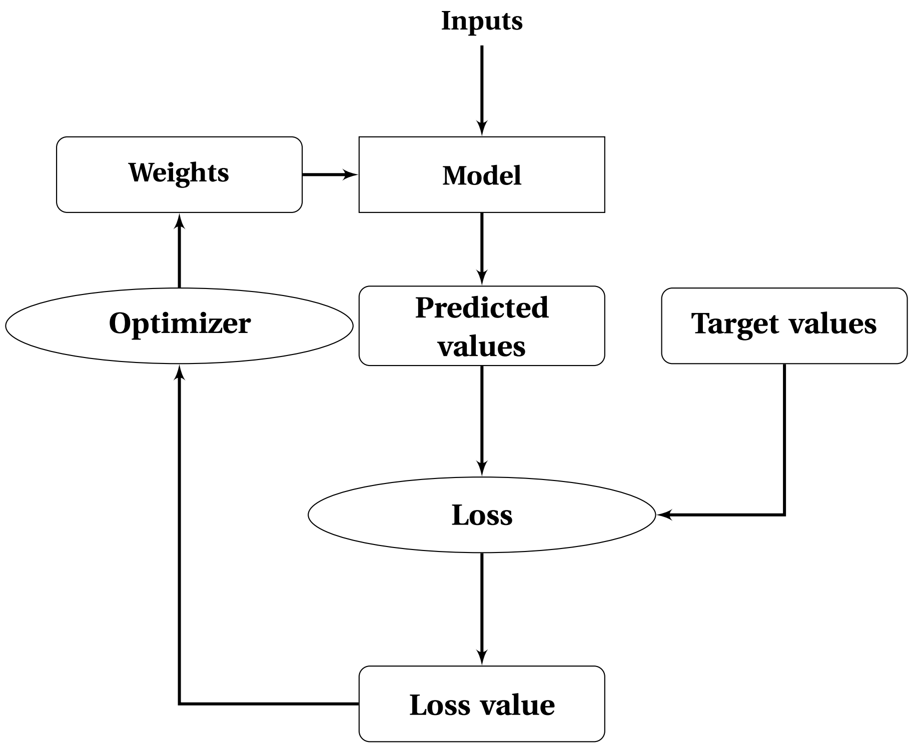


Figure 2.4: The training loop of a supervised machine learning model [39].

2.2.3.1 Loss

The loss function plays a crucial role in the training process of a neural network. Its purpose is to give a value that represents the difference between the true and predicted values. This value is then used by the optimizer to perform weight adjustments. The loss function needs to be differentiable for a gradient-based learning algorithm to work.

The selection of loss function depends on the task. The mathematical formulas of common loss functions are shown in table 2.2. The common loss function for regression tasks is the mean squared error (MSE). For classification either binary crossentropy or categorical crossentropy is used depending on the task. Binary crossentropy is used for tasks with two classes or in multi-label classification where the target value can have multiple classes simultaneously. Categorical crossentropy is used in multi-class classification where only one of the classes is the correct class.

2.2.3.2 Gradient Descent

The training of neural networks is based on gradient descent, which uses the gradient of the differentiable loss function to adjust the weights of the network. Gradient descent is an iterative optimization process during which steps are taken towards the minimum by following the negative direction of the gradient. Equation 2.3 shows the formula of gradient descent where new weights w_n are the computed from the old

Loss function	Usage	Formula
Mean squared error	Regression	$\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$
Binary crossentropy	Binary and multi-label classification	$-\frac{1}{n}\sum_{i=1}^{n}(y_n \cdot log(\hat{y}_n) + (1-y_n)\cdot log(1-\hat{y}_n))$
Categorical crossentropy	Multi-class classification	$-\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{k} y_j \cdot log(\hat{y}_j)$, k denotes the number of classes

Table 2.2: The formulas for loss functions. The loss is computed for predicted value y and target value $\hat{y}$ for n training samples.

weights w_{n-1} by subtracting the gradient ∇L multiplied by the step size ϵ. The gradient is the mean of all training samples. Figure 2.5 shows an example of the evolution of gradient descent in a two dimensional plane.

$$w_n = w_{n-1} - \epsilon \cdot \nabla L \tag{2.3}$$

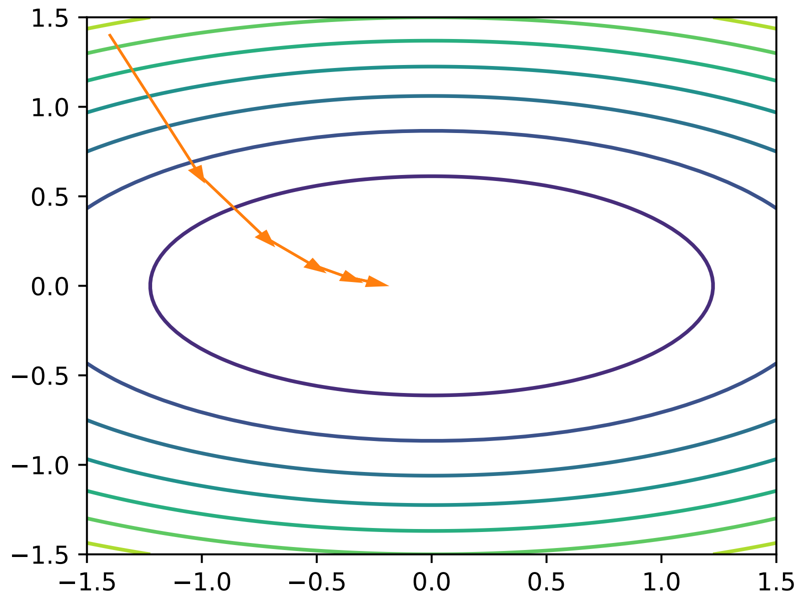


Figure 2.5: Gradient descent moving towards the minimum in a two dimensional plane.

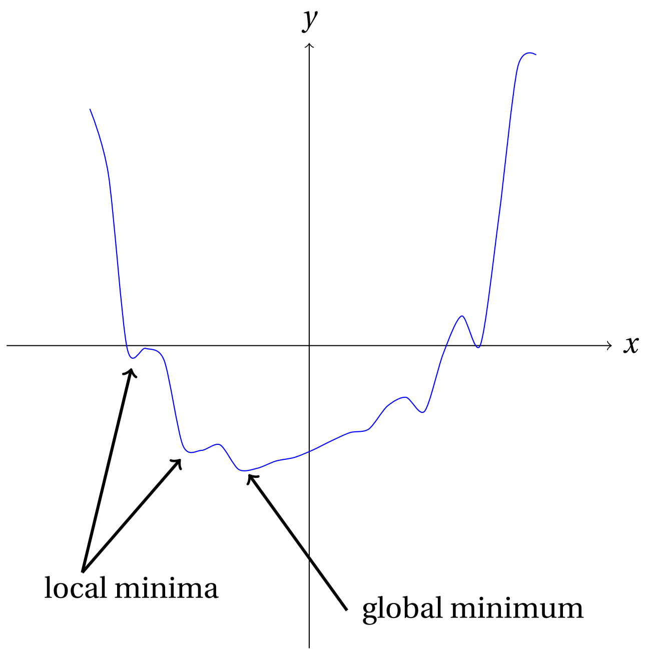


Figure 2.6: Function with local and global minimum.

2.2.3.3 Stochastic Gradient Descent

The original gradient descent algorithm, also known as batch gradient descent, performs one update for all training samples. This method of weight updates is prone to getting the minimization stuck in a local minimum instead of finding the global minimum when the error surface contains numerous local minima. An example of local and global minima is shown in 2.6. By performing weight updates for each training example, known as the stochastic gradient descent, the optimization process becomes noisier which has the benefit of bouncing out of local minima. This is advantageous when optimizing a multi-dimensional problem, such as modern machine learning models, which may contain hundreds of millions of trainable parameters, each of which is a dimension in the optimization space. Modern gradient based optimization methods use mini batches of tens to hundreds of training samples for weight updates instead of a single sample. In addition to the batch sizes, gradient descent can be improved by adapting the learning rate for better control of the step sizes as well as introducing momentum to better direct the gradient in multi-dimensional problems.

2.2.3.4 Adaptive Learning Rate

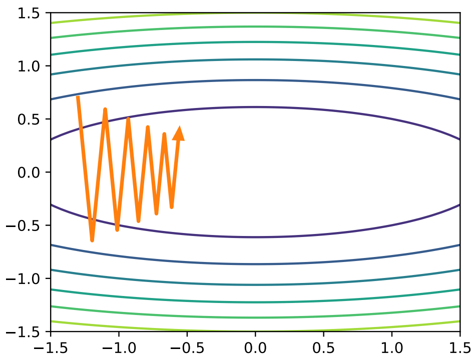
The step size ϵ, also referred to as the learning rate, controls the magnitude of the update performed by the gradient descent. By default the ϵ value is static for the optimization process. However this poses problems because a large ϵ helps the model to learn fast in the initial phase of the training, but causes oscillations around the minimum [46]. A small ϵ makes the learning process slow initially and can cause the model to be stuck in a local minimum. To mitigate these problems the learning rate ϵ can be adapted as the model approaches the global minimum, this way the model can learn fast in the initial phases using a larger ϵ and not bounce around global minimum in the later stages of the training.

2.2.3.5 Momentum

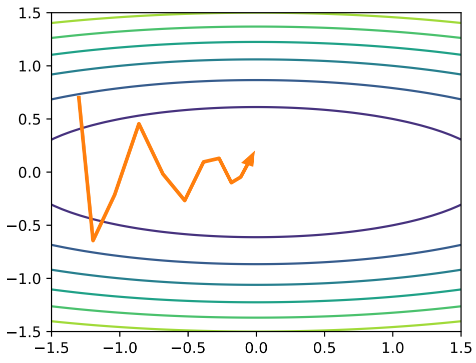
The error surface of the optimization problem often has a higher curvature in one dimension than another. These types of surfaces will cause gradient descent to oscillate instead of efficiently moving towards the minimum as shown in figure 2.7a. By introducing an additional momentum term z, referred to as velocity, these oscillations can be reduced. The effect of the velocity is controlled by the parameter $\gamma \in (0,1)$, which is often kept close to 1. The weight update becomes a two-staged process where first the new velocity z_n is computed based on the previous velocity z_{n-1}, shown in equation 2.4. The new velocity z_n is then used for the weight update, shown in equation 2.5. Figure 2.7b shows an example of gradient descent with momentum where the oscillations are significantly damped which speeds up the learning process. With equal amount of steps the gradient descent with momentum reaches closer to the minimum while gradient descent without momentum oscillates achieving smaller progress towards the minimum.

$$z_n = \gamma \cdot z_{n-1} - \epsilon \nabla L \tag{2.4}$$

$$w_n = w_{n-1} - z_n \tag{2.5}$$



(a) No momentum



(b) With momentum

Figure 2.7: Illustration of 10 steps taken by gradient descent without momentum and with momentum in a two dimensional plane with higher curvature in one dimension than another.

2.2.3.6 ADAM

The training process of neural networks can be further improved by more complex gradient based algorithms. Adaptive Moment Estimation (Adam) [47] optimizer is designed specifically for the use of optimizing deep learning models. Adam combines the advantages of two other popular optimizers: AdaGrad [48] which performs well with sparse gradients by adapting a per-parameter learning rate and RMSProp [49] which works well in non-stationary and online settings by utilizing the recent magnitudes of the gradients. The algorithm computes individual adaptive learning rates for different parameters using the estimates of first and second moments of the gradient. It also utilizes an exponentially decaying average of part gradients, similar to momentum as well as a decaying learning rate $\sqrt{t}$ where t is the time step of the training process. Reviews of different optimizers have shown that Adam is outperforming other optimizers in many deep learning tasks [46].

2.2.4 Backpropagation

The backpropagation algorithm [50] enables the efficient training of multi-layered neural networks. The weight adjustments for a single layer of neurons can be directly acquired from the loss function, but weight adjustments for a multi-layered network requires knowledge of the contribution of each weight in the intermediate layers to the loss. The backpropagation algorithm forms a composition function for multi-layer networks, as shown in figure 2.8, and computes the gradients for this composition function by using the chain rule of differential calculus. The algorithm is a special case of reverse-mode automatic differentiation [51].

The backpropagation algorithm consists of two parts: the forward and the backward pass. The forward pass is the computation of the network output with respect to its input. The loss value is then used for the weight adjustments. The backward pass starts from the output of the network, moving backwards in the network and computes the gradients of each layer with the chain rule to determine the contribution of each parameter to the loss value as shown in equation 2.6.

$$\begin{tikzcd} w \arrow[r, "f"] & x \arrow[r, "f"] & y \arrow[r, "f"] & z \end{tikzcd}$$

Figure 2.8: A computational graph that results in repeated subexpressions when computing the gradient. The same function f is applied at every step of a chain $x = f(w), y = f(x)$ and $z = f(y)$ [52].

$$\frac{\partial z}{\partial w} = \frac{\partial z}{\partial y}\frac{\partial y}{\partial x}\frac{\partial x}{\partial w} \tag{2.6}$$

2.3 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) [53] are designed to process time series data. This is achieved through parameter-sharing over each timestep of the time series. The same RNN operation is applied to each timestep and the state of the cell, commonly referred to as hidden state h, is used to encode the information from the past that is carried through time. The advantage of parameter-sharing is that the network does not need to learn all the rules for each position in the time-series like would be the case for a feed-forward network used for time series data. RNNs are feed-forward networks that add feedback loops for recurrence. To form a computational graph of a RNN the network is commonly unfolded as shown in figure 2.9. This unfolded graph provides an explicit description of the performed computations while also helping to illustrate the flow of information in a RNN [52].

A RNN layer can be applied to different types of sequences. The ways to apply a RNN to data are shown in figure 2.10. The RNN that produces an output for each input is referred to a many-to-many RNN also known as a sequence-to-sequence model. The second type produces a single output after processing the full sequence and is referred to as many-to-one. The third type uses the same input for each timestep to produce different output for each timestep and is referred to as one-to-many. Different types of RNN layers can be chained, for example using a many-to-one RNN followed by one-to-many RNN can be used to form an autoencoder [54].

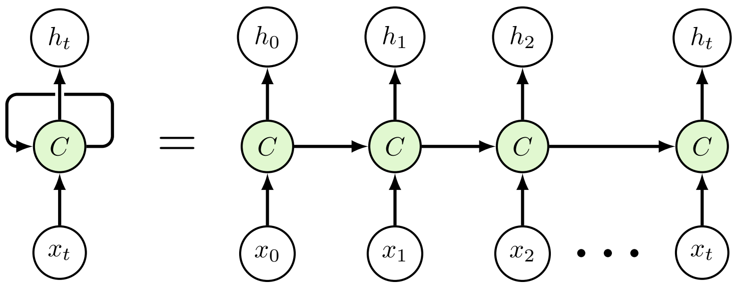


Figure 2.9: RNN operation C is applied for each timestep with input x_t to produce state h_t and can be unrolled to provide an explicit description of the performed computations.

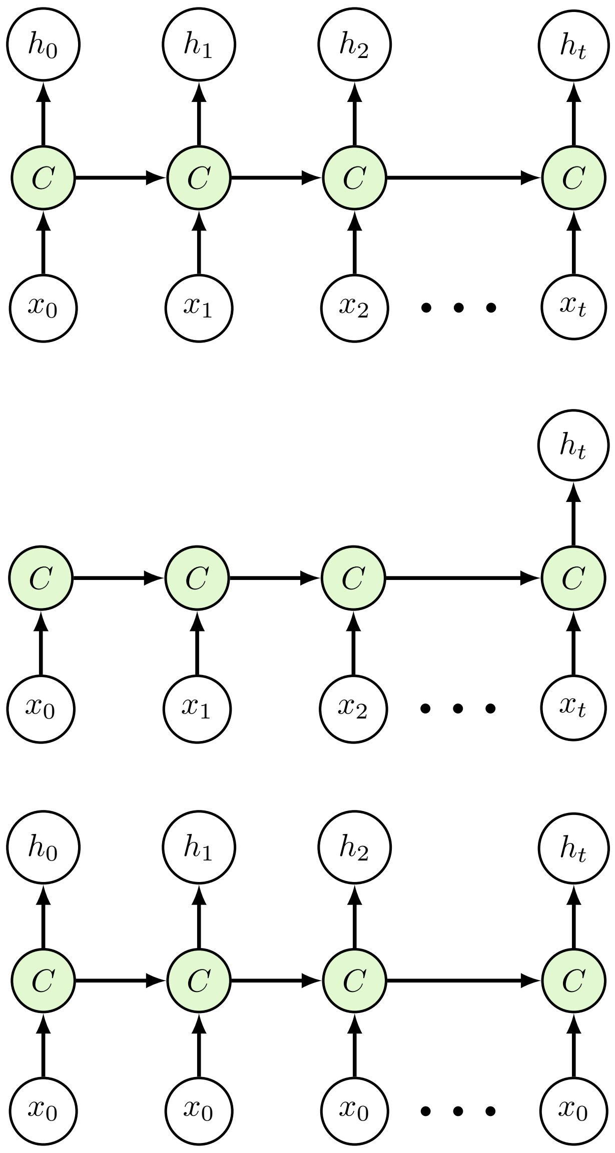


Figure 2.10: RNN operation for different input and output types. (top) Multiple inputs, multiple outputs known as many-to-many, (middle) multiple inputs and single output known as many-to-one, (bottom) single input with multiple outputs known as one-to-many.

2.3.1 Simple RNN

The simple RNN structure [55] is the simplest RNN type that consists of two weight matrices. Equation 2.7 shows the operations for RNN cell for each timestep. The weight matrices in the cell are W which is used for the input x_t which is a $n \times m$ square matrix. n is the unit count and m is the number of input features for each timestep of x_t. The weight matrix U is used for the hidden state h_{t-1} and is a $n \times n$ square matrix.

The bias vector is of length n. The output of the RNN is a vector of the size n. The activation in the original publication is tanh, but in this work the ReLU because its simplicity makes it ideal for firmware implementation.

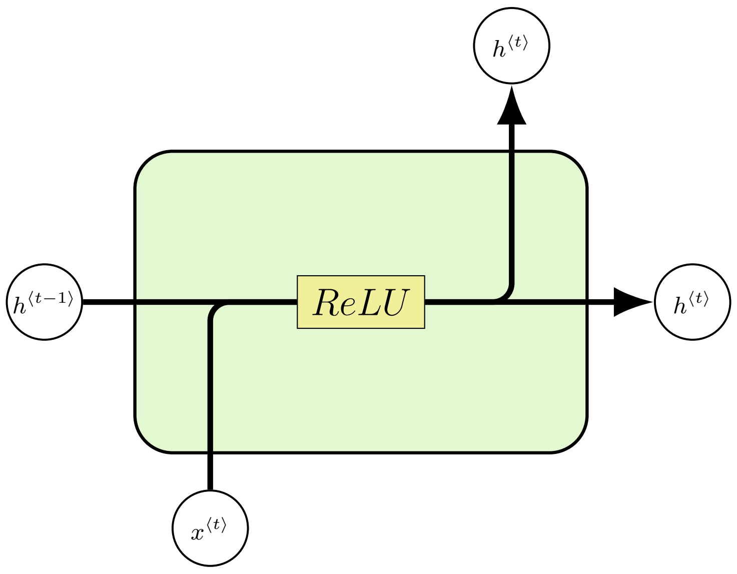


Figure 2.11: RNN operation for previous hidden state h_{t-1} and input x_t using a ReLU activation produces output h_t.

$$h_t = activation(Wx_t + Uh_{t-1} + b) \tag{2.7}$$

2.3.2 Backpropagation Through Time

RNNs are trained using the backpropagation through time algorithm (BPTT) introduced in [56] which is a generalization of the backpropagation algorithm. It works by unfolding the RNN: every time step has its own input but the parameters are shared for all time steps. The backpropagation algorithm is then used for the gradients of the RNN.

2.3.3 Long Short-term Memory

Long Short-Term Memory (LSTM) [57] is an evolution of the simple RNN structure designed to process complex information through time. Better handling of information through time is achieved through a gated architecture with four internal gates (forget gate f, input gate i, gate for the cell state candidate $\tilde{c}$ and output gate o) and by carrying two states through time: the hidden state h and the cell state c. This gated architecture has been proven to perform better on longer sequences by allowing it to learn to selectively access the states and to mitigate the vanishing gradient problem

[58]. The addition of the forget gate plays an essential role in enabling the LSTM to perform well over long sequences and it can be applied to infinite input streams [59].

The computation of LSTM for time step t consists of computing the gates using the input x_t and previous hidden state h_{t-1} as shown in equations 2.8, 2.9, 2.10 and 2.11. The cell state c_t is acquired summing the element-wise product of previous cell state c_{t-1} and forget gate output f_t with element-wise product of input gate output i_t and the cell state candidate $\tilde{c}_t$ as shown in equation 2.12. The hidden state h_t is acquired by performing an element-wise product of the output gate o_t and the cell state c_t passed through a tanh activation function.

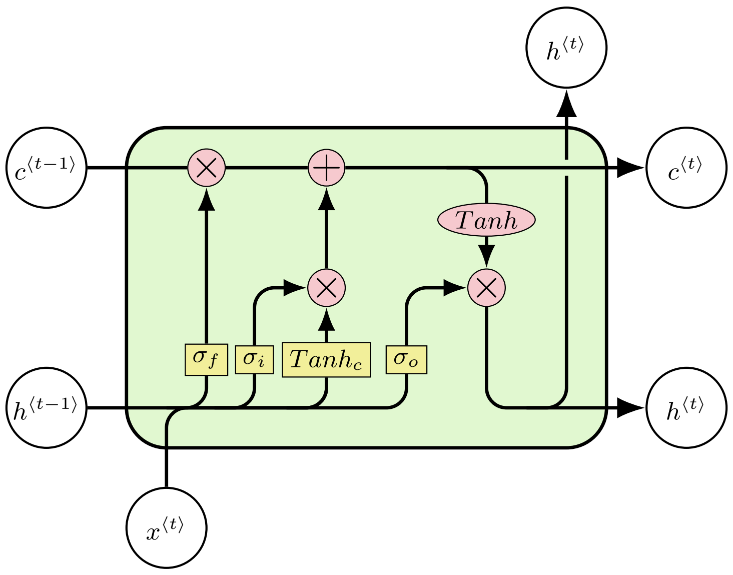


Figure 2.12: LSTM cell structure. For each timestep t the cell uses the past hidden state h_{t-1} and cell state c_{t-1} as well as input x_t. Outputs are the hidden state h_t and cell state c_t.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f) \tag{2.8}$$

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{2.9}$$

$$\tilde{c}_t = tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{2.10}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o) \tag{2.11}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \tag{2.12}$$

$$h_t = o_t \odot tanh(c_t) \tag{2.13}$$

3 Energy Reconstruction in the LAr Calorimeter

Summary

3.1 Introduction

This chapter evaluates the performance of RNNs for the energy reconstruction in the LAr calorimeter. The size of the network as well as amount of samples required for accurate energy reconstruction are evaluated. In addition, the selection of the optimizer and model criteria to pick the best performing model are evaluated. The RNN can be applied in a many-to-one way, which corresponds to a sliding-window application in section 3.3. The other possible way to apply the RNN is in a many-to-many way which is referred to as single-cell in section 3.4. The RNN is also extended to reconstruct the phase shift of the electronic pulse in addition to the amplitude of the pulse in section 3.5. The ability of the RNNs to perform over a range of instantaneous luminosities is evaluated in section 3.6.

3.2 Dataset

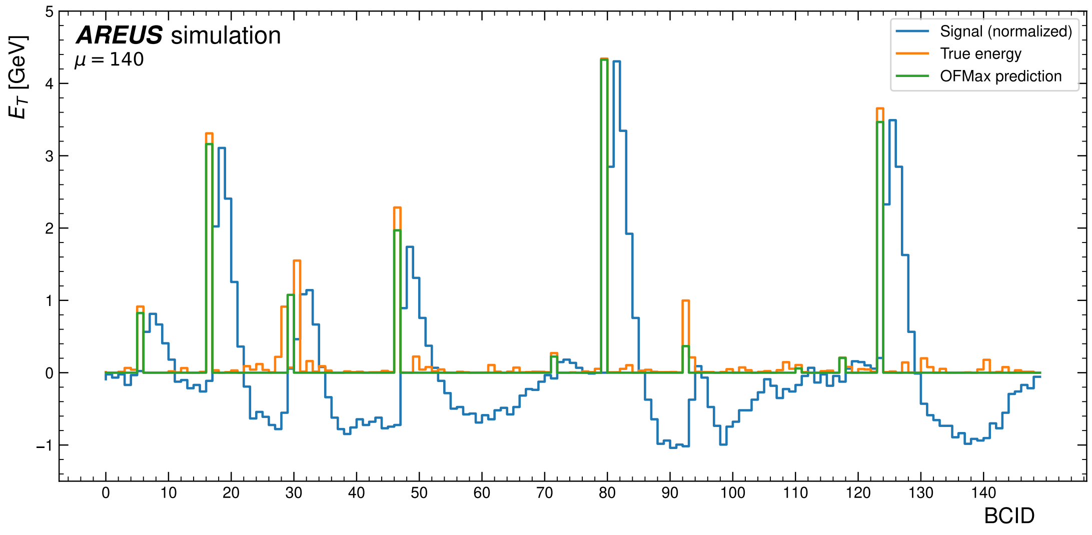


Figure 3.1: An example sequence with 150 bunch crossings that contains isolated and overlapping signals. The true energy deposits as well as the energy reconstructed by OFMax are also shown. The sample amplitude is normalized to the deposited energy in GeV.

The dataset used to train and evaluate the performance of the neural networks contains 2 million simulated continuous bunch crossings created using AREUS with an average pileup of $\mu = 140$. The simulation is done for a calorimeter cell in the middle layer of the barrel (labeled EMB middle) at $\eta = 0.5125$ and $\phi = 0.0125$. In addition to pileup, a signal of up to 5 GeV is injected at a random interval with a mean of 30 bunch crossing and standard deviation of 10. This results in a dataset that contains a significant amount of overlapping events where either two peaks are overlapping or the second peak is in the tail of the previous pulse. The dataset is the same as the one used in [1]. Figure 3.1 shows an example sequence that contains overlapping signals where the pulse height is affected by the previous pulses. The deposits that are affected by the tail of the previous pulse, such as around BCID 50 and 95, lead to the energy being underestimated by optimal filtering with maxfinder (OFMax), explained in section 1.4.3.1. The largest misprediction happens around BCID 30 where the tail of the pulse has two deposits within one bunch crossing from each other. This leads OFMax to predict a non-zero energy at the BC which is between the two deposits. The energy is set to 0 at the correct BCs. OFMax performs well in the case of isolated pulses, such as the deposits around BCID 15, 80 and 125.

3.3 Energy reconstruction with Recurrent Neural Networks in the sliding window architecture

In the sliding window architecture, the LAr sample sequence is divided into relatively short overlapping sub-sequences with each sub-sequence corresponding to one energy output following the many-to-one RNN architecture explained in chapter 2.3. The target energy for each sub-sequence is chosen so that the sequence contains samples both before and after the deposited energy. The samples in the past allow to correct for the effects from out-of-time pileup. The samples after the deposit probe the resulting pulse amplitude to reconstruct the energy. The length of the sub-sequence has to be chosen to achieve an optimal performance of the networks while keeping in mind that each added sample increases the resource usage in the FPGA. Moreover, samples added after the deposited energy will increase the total reconstruction latency since the computation of the energy will need to wait for these samples to be available.

The chosen architecture of the network consists of one RNN layer followed by a single neuron for the output layer to transform the vector output of the RNN into a single value corresponding to the energy. Four samples following the energy deposit are used to probe the peak of the pulse and are found to be sufficient to reconstruct the pulse amplitude. Further samples are used for out-of-time pileup corrections. Figure 3.2 depicts the sliding window application of the RNN with one sample prior to the energy deposit totaling 5 samples as input to the neural network.

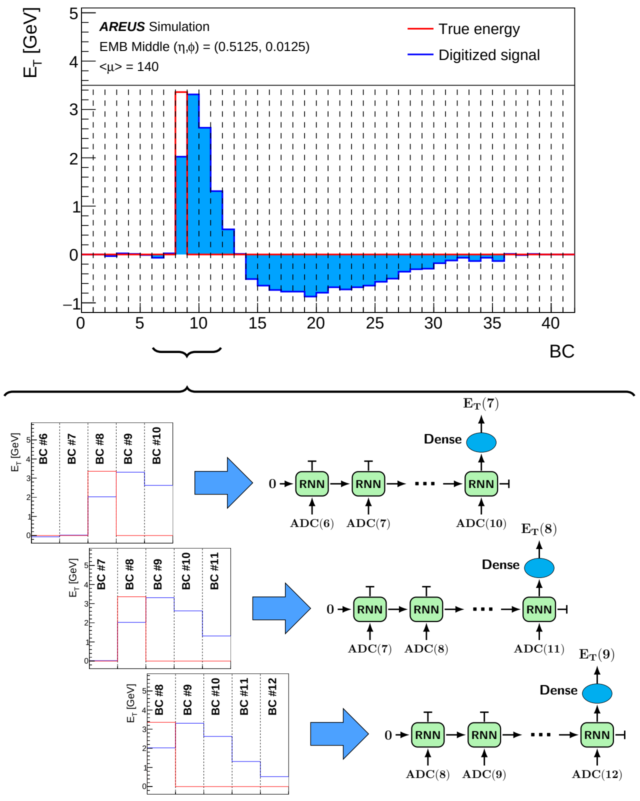


Figure 3.2: Sliding window application of the RNN, where a window of fixed size is moved across the sequence to process it in smaller, overlapping segments, allowing for the reconstruction of the deposited energy and the correction of the effect of out-of-time pileup [1].

3.3.1 Optimizing RNNs for LAr

This section evaluates the choice of RNN hyper-parameters for the energy reconstruction. The goal is to find the best performing network that will fit on the FPGA. The network parameters and the choice of the optimizer are evaluated to select the best performing network. The MSE loss function is widely used for regression tasks and is chosen for all the training performed in this work. The performance of the networks is examined using several criteria as explained in section 3.3.1.1.

3.3.1.1 Model selection criteria

The neural network training is a stochastic process with the network weights initialized with random values. In addition, the training data is also randomly shuffled between each epoch. The common initialization for RNNs is glorot uniform [60] for the weight matrix, orthogonal for the recurrent weight matrix and zeros for the biases.

Several networks with the same configuration but different initializations are trained and the best performing is selected. However a selection criteria is needed to evaluate which one of the models performs the best. The model needs to accurately reconstruct higher energies but also suppress the bunch crossings that contain only noise.

The comparison of the models is done using either RMSE shown in equation 3.1 where E_T^{pred} is the predicted value, E_T^{true} is the true value and n the number of samples or standard deviation shown in equation 3.2 where E_T^{pred} is the predicted value, E_T^{true} is the true value, $\bar{x}$ is the mean of the predicted values and n is the number of samples.

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum (E_T^{pred} - E_T^{true})^2} \tag{3.1}$$

$$\sigma = \sqrt{\frac{\sum((E_T^{pred} - E_T^{true}) - \bar{x})^2}{n}} \tag{3.2}$$

The models can either be evaluated for all events, this is mostly dominated by low energy events, or only for events above a selected threshold. The evaluation for high energy reconstruction performance is done only for events above 240 MeV. The cut at 240 MeV is chosen as it corresponds to 3σ above noise level for the chosen calorimeter cell. The different selection criteria probe the performance in different regions and do not necessarily lead to the same best model. Table 3.1 shows the RMSE and the standard deviation for the best models selected by the different selection criteria. Among 35 trained RNNs with 8 units and a sequence length of 5, the RMSE and standard deviation criteria on all events lead to the same best model while different best models are selected if a cut on $E_{true}^T > 240$ MeV is applied. The three selected models have only minor differences (<1 MeV) in performance. Figure 3.3 shows the performance of the three models for the two energy ranges $E_{true}^T \leq 240$ MeV and $E_{true}^T > 240$ MeV.

Selection criteria	RMSE/Stdev	Stdev 240	MSE 240
σ	**49 MeV**	51 MeV	50 MeV
$\sigma, E_T^{true} > 240 MeV$	139 MeV	**135 MeV**	139 MeV
RMSE	**49 MeV**	52 MeV	50 MeV
RMSE $E_T^{true} > 240 MeV$	150 MeV	149 MeV	**148 MeV**

Table 3.1: Model performance for different model selection criteria. Each column represents the selected model and rows represent the corresponding performance of the model with that criteria. The numbers highlighted in bold represent the model that has the best performance for a given criteria.

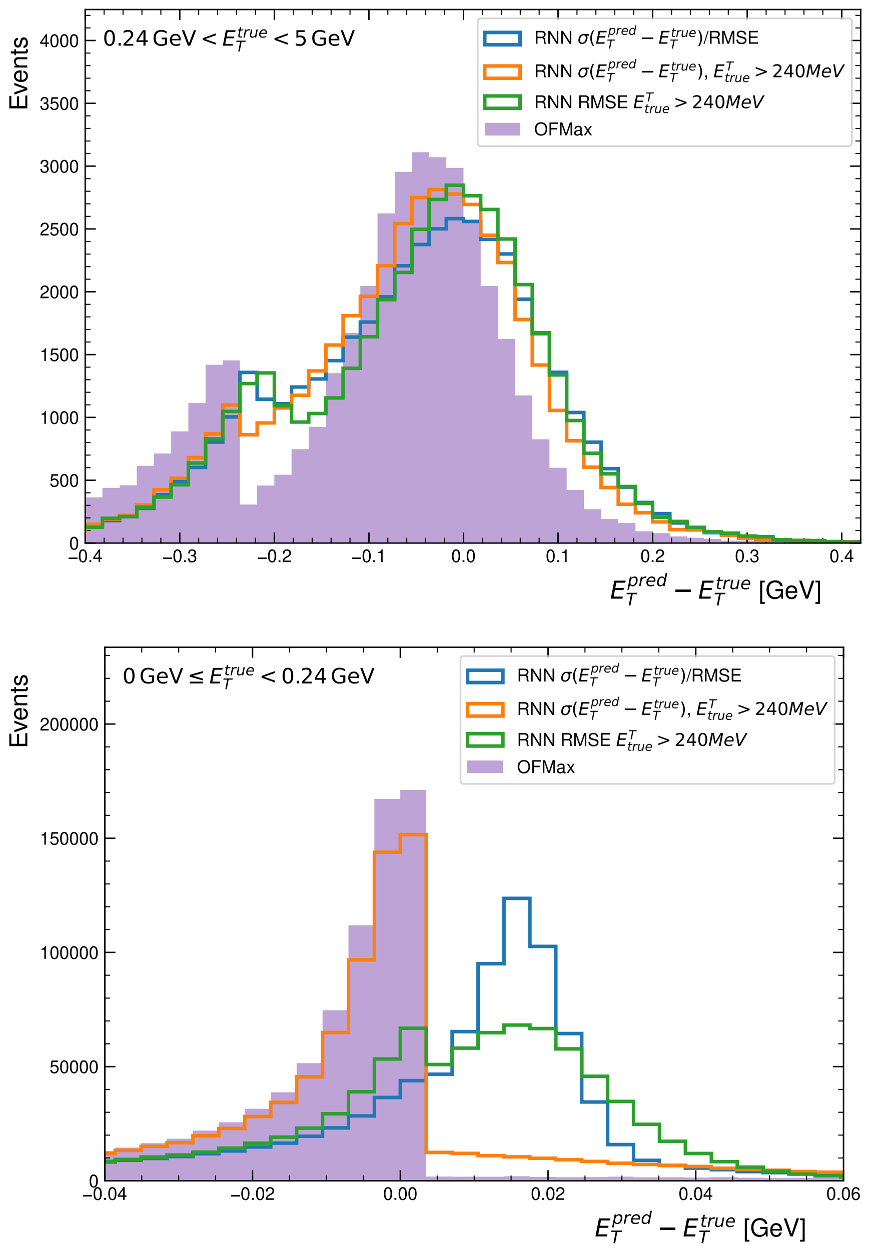


Figure 3.3: Transverse energy resolution for the three RNN models selected with different model selection criteria in comparison with OFMax for (top) $E_T^{true} > 240$ MeV and (bottom) $E_T^{true} < 240$ MeV.

3.3.1.2 Selection of the optimizer

The choice of the optimizer plays a crucial role in training neural networks. Different optimizers yield vastly different results. The optimizers evaluated are: SGD, SGD with momentum of 0.9, RMSprop and Adam. Figure 3.4 shows the validation loss in RMSE for the models using different optimizers for an RNN with 8 internal units. 20 different trainings with different initialization is used for each optimizer. The mean of these 20 trainings is shown in solid line while the bands show the range between the minimum and maximum values.

Table 3.2 shows the rate of convergence and the performance of the optimizers. Overall the Adam optimizer performs the best with the highest amount of trainings resulting in a performance better than OFMax. The dead neuron effect with the ReLU activation function is the cause of some initializations not converging. However all models that converged using the Adam optimizer perform better than OFMax. RMSprop is the second most efficient optimizer with equal percentage of converged networks as with Adam. However the performance of the trained networks is not as good as with Adam with only 40% of them being better than OFMax. RMSprop also shows less stable training with heavy oscillations after several epochs of training. Networks trained using SGD achieve poor performance, none of the models perform better than OFMax after 20 epochs. SGD with momentum improves the performance with 55% of the trainings having overall performance better than OFMax.

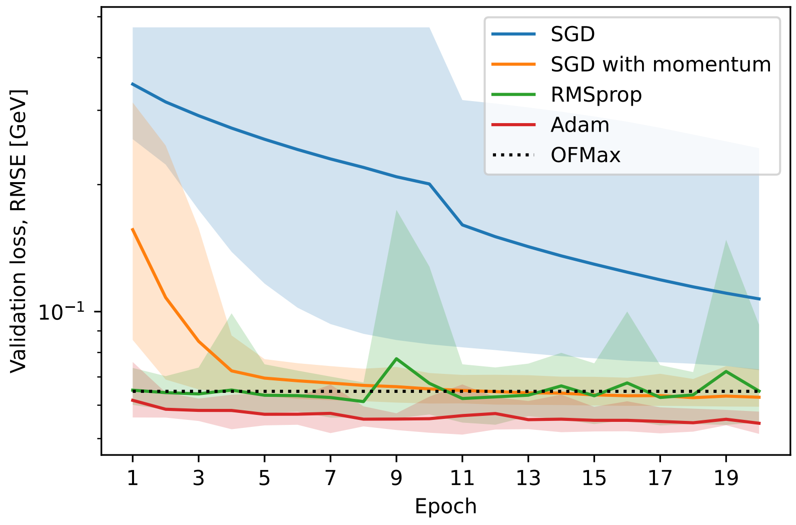


Figure 3.4: The effect of the optimizer for the converged models. Each model was trained for 20 epochs. Mean RMSE on the validation set is shown in solid line and the shaded area shows the minimum and maximum RMSE values for that epoch. The RMSE of OFMax is shown with the dashed line.

Optimizer	**SGD**	**SGD with momentum**	**RMSprop**	**Adam**
Converged	60%	55%	80%	60%
$RMSE_{\text{RNN}} < RMSE_{\text{OFMax}}$	0%	45%	55%	60%
$\sigma_{\text{RNN}} < \sigma_{\text{OFMax}}(E_T^t rue > 240\text{ MeV})$	0%	50%	55%	60%

Table 3.2: Convergence and performance after training for 20 epochs 8 unit RNN with different optimizers. 20 independent trainings with different random initializations are used for each optimizer. The percentage of converged models, the fraction of models with better RMSE than OFMax, and the fraction of model with a standard deviation better than OFMax for energy deposits above 240 MeV are shown.

3.3.2 Energy reconstruction performance

The performance of RNNs in energy reconstruction is affected by several factors. One such factor is the size of the RNN network in terms of the RNN unit count. The network size determines the capacity of the network to learn complex patterns in the input signal and can have a significant impact on its ability to generalize to unseen data. Another major factor is the sequence length used, which affects both out-of-time pileup correction as well as the reconstruction of the amplitude. The optimization of these factors is crucial to achieve the best possible energy reconstruction performance using RNNs. The evaluated networks are the simple RNN as well as the more complex LSTM.

3.3.2.1 Optimizing the neural network size for simple RNN architecture

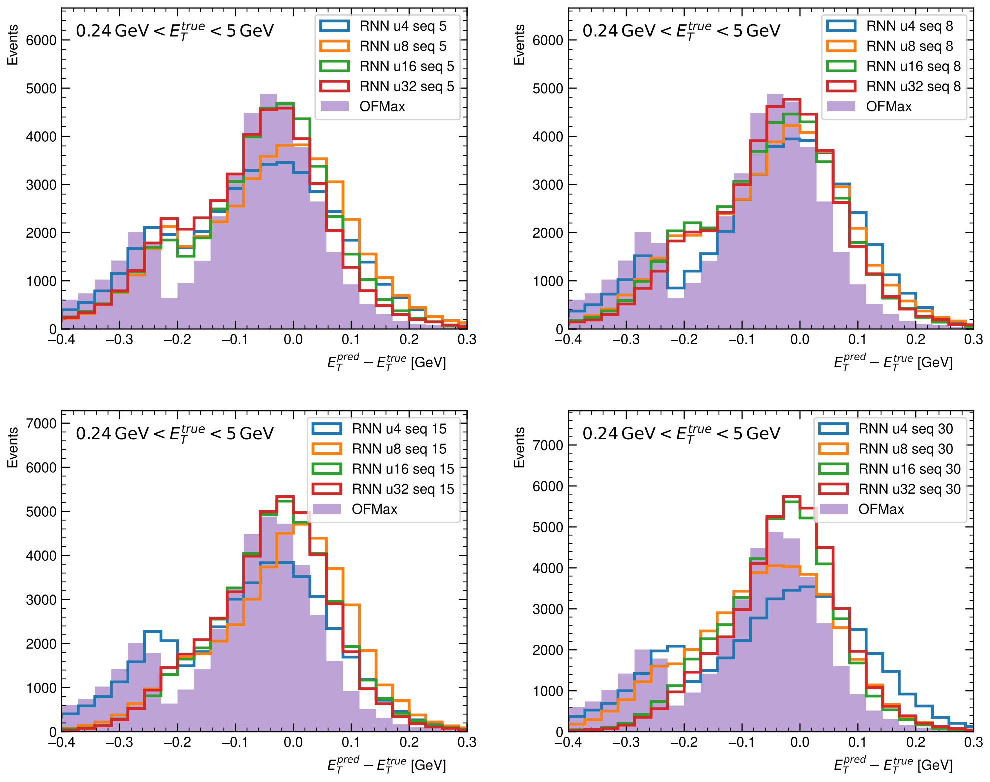


Figure 3.5: Transverse energy resolution for different simple RNNs compared to OFMax. Each plot has four RNNs with different unit size (u) ranging from 4 to 32 while the sequence length (seq) is kept unchanged.

The two main model hyper-parameters for RNNs are the unit size and the input sequence length. These two parameters are optimized to ensure a good energy resolution both for overlapping and isolated pulses. The training of the networks utilizes two tools in addition to the optimizer: learning rate reducer to lower the learning rate when the validation loss is stagnating and early stopping to stop the training when reducing learning rate does not help reduce the validation loss. Figure 3.5 shows the effect of the network size for four different sequence lengths for simple RNN evaluated for four network sizes. The OFMax distribution shows two peaks: the peak close to 0 corresponds mainly to isolated pulses where OFMax performs relatively well, the second peak at lower values correspond to overlapping pulses where OFMax predicts low or 0 energy.

For a sequence length of 5, which has only one sample before the energy deposit, none of the networks are able to correct well for out-of-time pileup that correspond to the second peak at low values. Larger network is able to improve in the core of the distribution. Increasing the sequence length to 8 with 4 samples in the past shows small improvement in energy resolution. All networks show improvement when using longer sequence length. With a sequence length of 15 the second peak is almost gone for all networks except the smallest which has 4 units. With this sequence length the 16 and 32 unit RNNs are also able to achieve better performance than OFMax in the main peak. When increasing the sequence length to 30, which corresponds to the full length of the LAr electronic pulse, the two larger networks are able to almost fully mitigate the second peak. However the two smallest networks with 4 and 8 units are not capable of probe all the information from the 30 samples and thus do not correct the second peak.

3.3.2.2 Optimizing neural network size for LSTM architecture

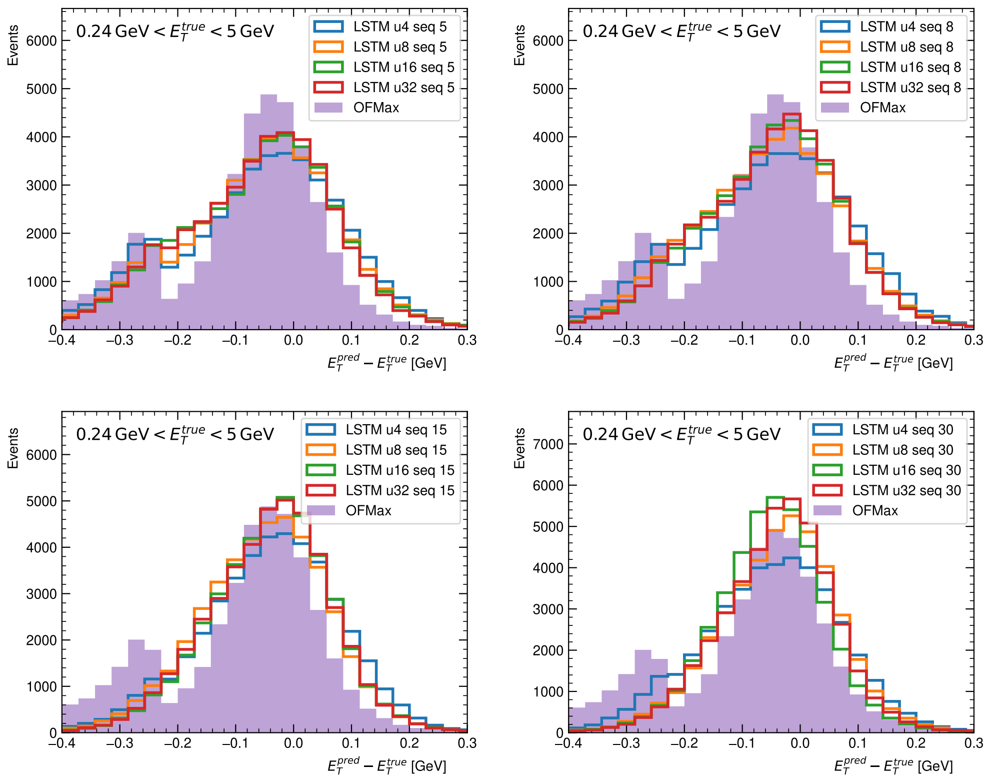


Figure 3.6: Transverse energy resolution for different LSTM compared to OFMax. Each plot has four LSTMs with different unit size (u) ranging from 4 to 32 while the sequence length (seq) is kept unchanged.

The LSTM network has more parameters than the simple RNN. Figure 3.6 shows the effect of network size for four different sequence lengths for LSTM. For sequence length of 5, similarly to simple RNN, none of the networks achieve better performance in the main peak than OFMax. However OFMax shows a much larger second peak so that its overall resolution in terms of standard deviation is not as good as with the LSTMs. With a sequence length of 8 the LSTM networks have increased performance with a higher peak in the main part and bringing the secondary peak closer to zero. A sequence length of 15 allows the LSTM networks to greatly improves both the main peak as well as the secondary. The LSTMs with 8, 16 and 32 units are also able to achieve better performance than OFMax in the main peak. While the smallest LSTM improves slightly in the main peak, it is able to well mitigate the second peak. A sequence length of 30 allows the three larger networks to mitigate the second peak as well as to have better performance in the main peak. However this sequence length is also too long for the smallest network with 4 units which cannot outperform OFMax.

3.3.2.3 Summary of the network performance as a function of the sequence length

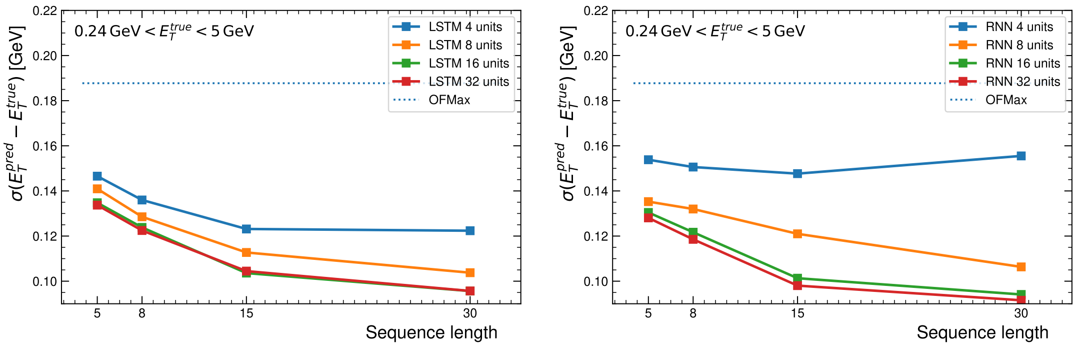


Figure 3.7: Standard deviation of $E_T^{pred} - E_T^{true}$ as a function of the sequence length for (left) LSTM and (right) simple RNNs with different internal unit sizes.

To fit the stringent requirements in terms of latency and FPGA occupancy, it is essential that the network is as small as possible. Figure 3.7 shows the simple RNN and LSTM performance as a function of the sequence length. As expected the longest sequence length of 30 has the best performance but it requires 6 times more computations than the smallest sequence length of 5. All LSTM networks benefit from longer sequence length, however, only the 3 biggest RNNs have the same benefit. In fact the RNN with only 4 units is not able to take advantage of the longer sequence length which is due to its reduced internal neural network size. For both types of networks one can see an elbow structure at a sequence length of 15 where doubling the sequence length to 30 does not bring significant improvement in the performance.

3.3.2.4 Summary of the network performance as a function of the unit count

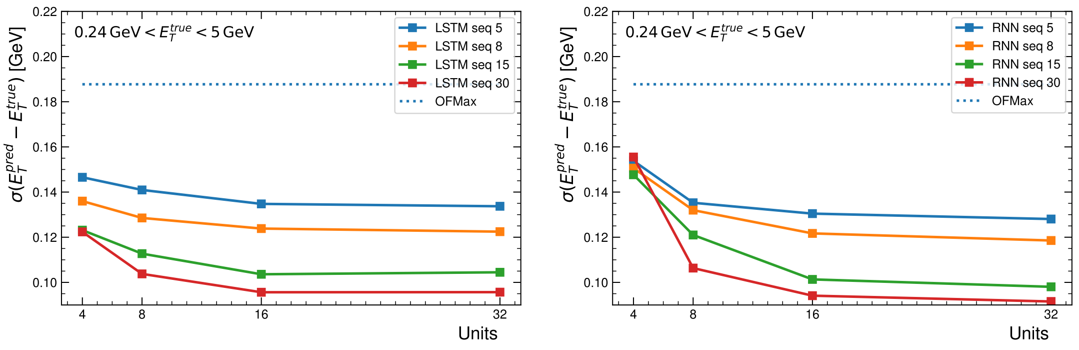


Figure 3.8: Standard deviation of $E_T^{pred} - E_T^{true}$ as a function of the sequence length for (left) LSTM and (right) simple RNNs with different sequence lengths.

The number of parameters increases quadratically with the unit count. It is essential to keep the unit count as low as possible. Figure 3.8 shows the LSTM and simple RNN performance as a function of RNN unit count for different sequence lengths. Adding units improves the performance but the effect is small beyond 16 units where one can see a convergence of all networks above this value. One can also notice that the simple RNN, due to its smaller size, benefits more from the increase of the unit size.

3.3.2.5 Energy resolution as a function of the parameter count.

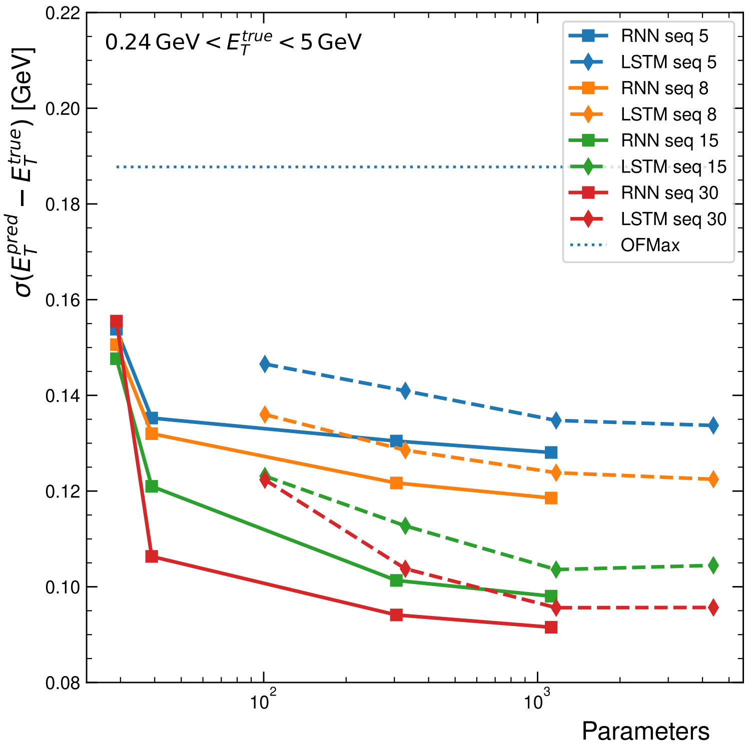


Figure 3.9: Performance as a function of the parameter count for LSTM and RNN

When selecting the model for the real-time energy reconstruction, minimizing the parameter count plays a crucial role as the resources are limited. Figure 3.9 shows the resolution as a function of the number of parameters of different neural networks. The LSTMs have a higher number of parameters and start with a better performance at a low number of units. However with the increased number of units the simple RNNs outperform the LSTM. Overall the simple RNNs perform better for the same number of parameters compared to the LSTM.

3.3.2.6 Performance as a function of the time gap between two subsequent pulses

The HL-LHC collision rate of up to 200 simultaneous proton-proton collisions will present a challenge for energy reconstruction by the large in-time pileup and because of the pulse response of the detector, which affects the energy reconstruction for up to 25 subsequent bunch crossings leading to out-of-time pileup. In addition to high pileup, the Phase-II triggering scheme has no dead-time which requires that the energy reconstruction algorithm is able to reconstruct accurately deposits that happen within a few bunch crossings of each other as well as the ability to assign the energy for the correct bunch crossing. This increases the importance of properly reconstructing the energy of overlapping pulses. Figure 3.10 shows the energy resolution as a function of the time gap between two high energy deposits ($E_T^{true} > 240$ MeV) for OFMax. This plot shows that the OFMax performance is greatly reduced in case of overlapping pulses with a time gap smaller than 20 bunch crossings.

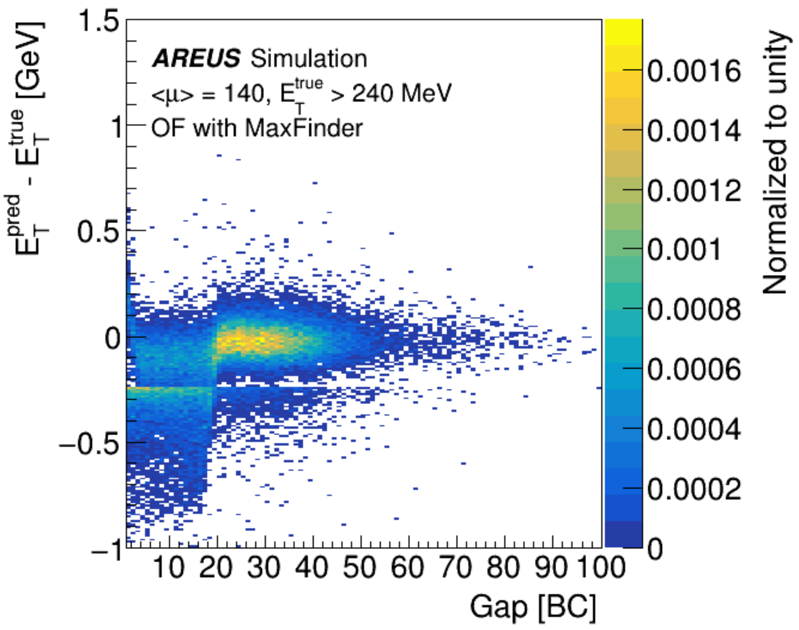


Figure 3.10: OFMax energy resolution as a function of the gap between two energy deposits for higher energies than 240 MeV.

RNNs are able to take advantage of samples before the deposit to correct for out-of-time pileup. When using a sequence length of five, the RNN is able to partially correct the effect of overlapping pulses as shown in figure 3.11 for simple RNN and in figure 3.12 for LSTM. The correction is possible since the input sequence contains one sample from before the peak which enables the RNN to capture some of the out-of-time pileup in the signal. However, the correction is limited since only one sample is not enough to capture all the relevant information for the RNN to make accurate predictions. This insufficient amount of past data is the reason for the undershoot in the gap region of up to 20 and the overshoot around the gap region of 20. This overshoot is due to edge effects where only the sample before the deposit is affected by previous energy deposits and not the actual pulse that the network is trying to probe. Nevertheless, even a small correction can lead to significant improvements in the overall energy resolution as shown in previous sections where the overall energy resolution was better than for OFMax. Increasing the size of the network by increasing the number of units improves mainly the resolution for isolated pulses with a gap above 20 bunch crossings.

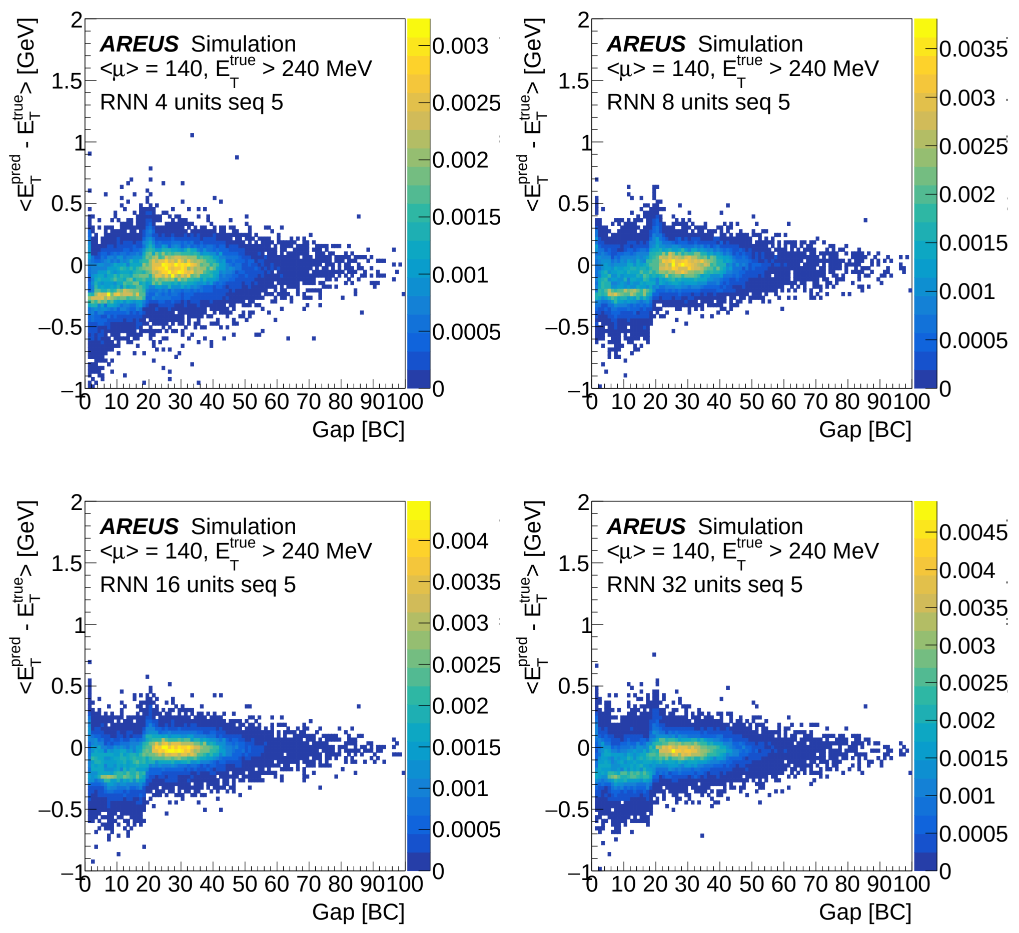


Figure 3.11: Resolution as a function of the gap between two energy deposits for simple RNNs with a sequence length of 5 and different unit sizes.

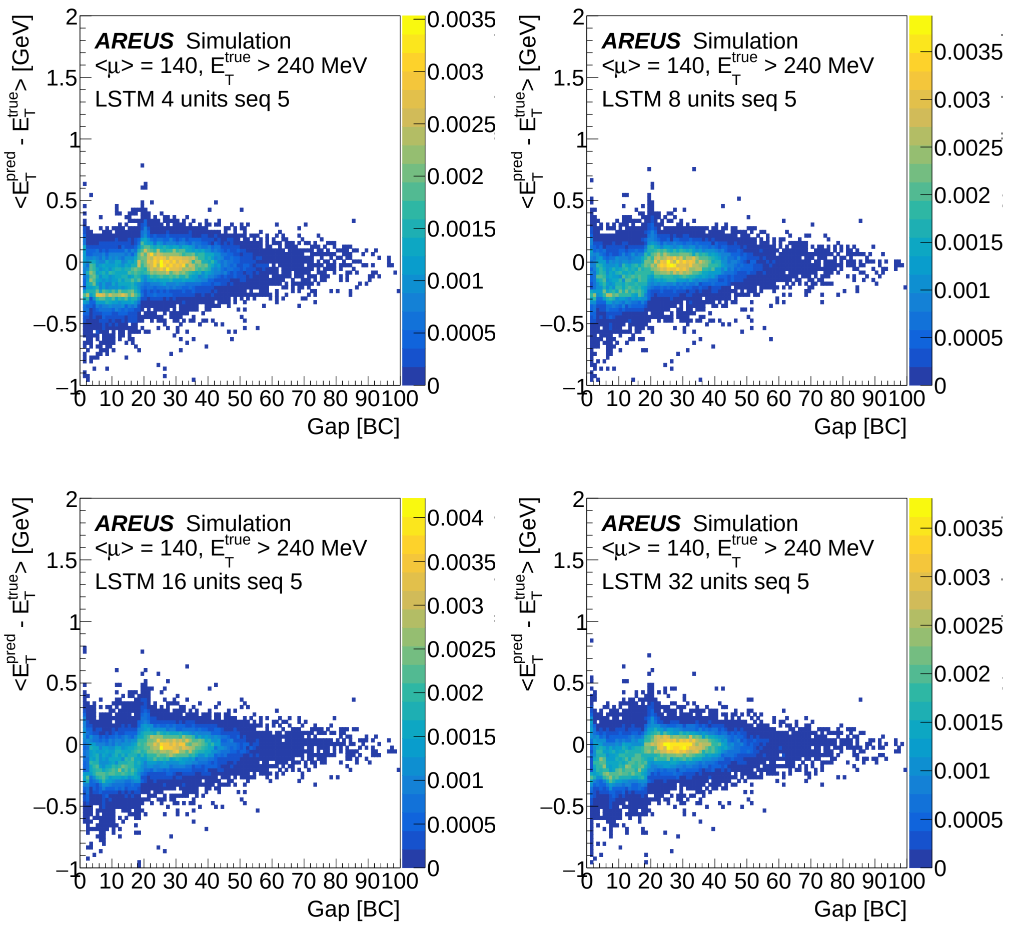


Figure 3.12: Resolution as a function of the gap between two energy deposits for LSTMs with a sequence length of 5 and different unit sizes.

When increasing the sequence length to eight, the RNN is able to correct for the overlapping events slightly more effectively, especially in cases where the time gap between two overlapping pulses is around 8 BCs as shown in figure 3.13 for simple RNN and in 3.14 for LSTM. This is because a longer sequence length allows the RNN to have more information about the past events, which is crucial for correcting for out-of-time pileup. The longer sequence length also allows the RNN to better distinguish between overlapping peaks and better attribute the energy to the appropriate peak appears which is seen in the figures where there are fewer events at the $E_T^{pred} - E_T^{true} = -240$ MeV region in the plots in comparison to those with shorter sequence length. However, the correction is still fairly minimal, and in most cases the RNNs are not able to fully correct for overlapping events.

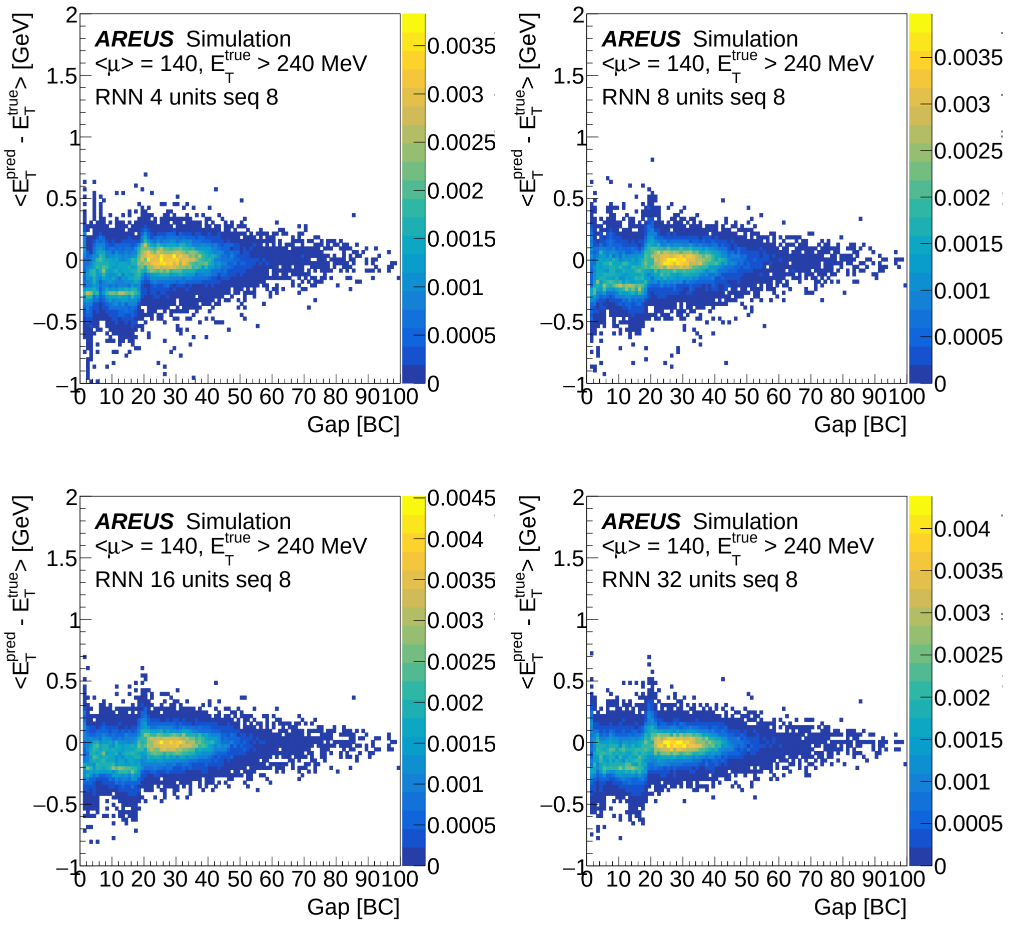


Figure 3.13: Resolution as a function of the gap between two energy deposits for simple RNNs with a sequence length of 8 and different unit sizes.

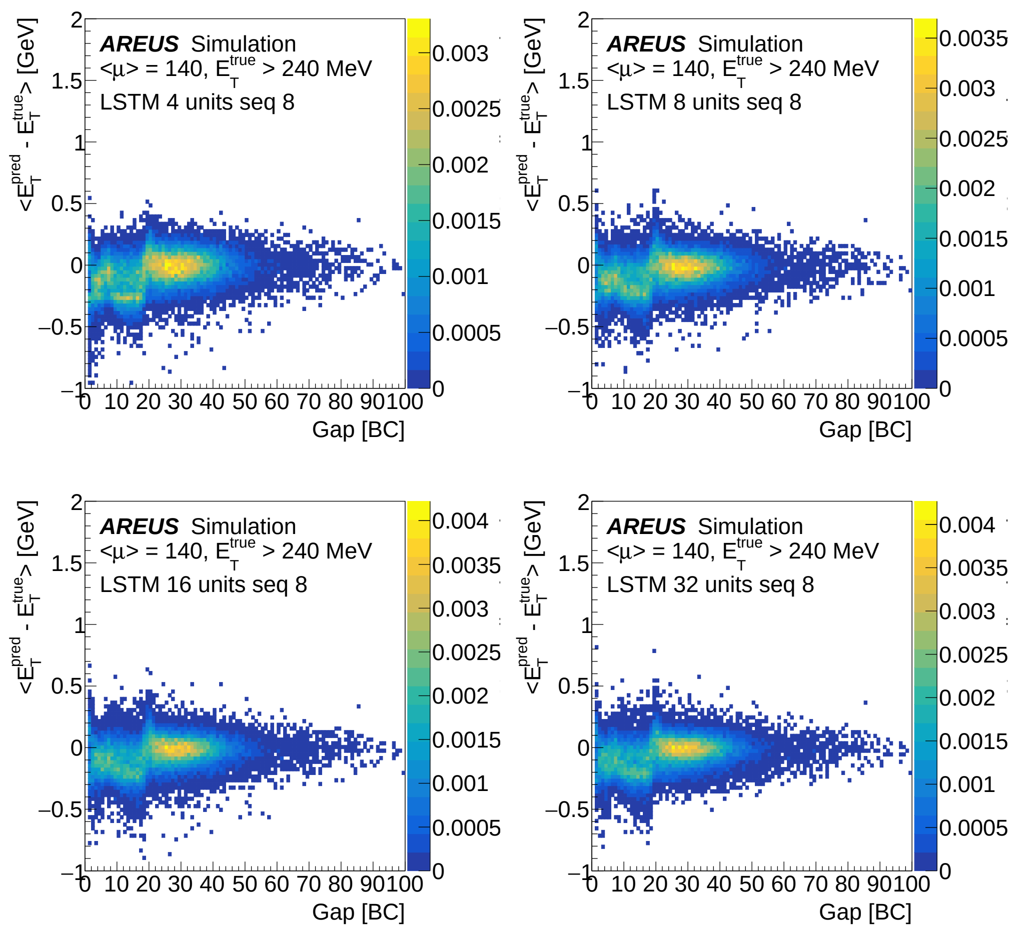


Figure 3.14: Resolution as a function of the gap between two energy deposits for LSTMs with a sequence length of 8 and different unit sizes.

Using a sequence length of 15, shown in figure 3.15 for simple RNN and in 3.16 for LSTM, the RNNs are able to properly reconstruct the energy for overlapping pulses as the sequence length is enough to infer the conditions before the deposit. However this is mainly true for the networks that are large enough to be able to carry the information in the internal state through the full sequence. This results in a significant improvement in the overall energy resolution for networks that have 8, 16 or 32 units. While the energy resolution is a lot better than with shorter sequence lengths, there are still cases where the RNN is not able to fully correct for the past events especially in the region around a time gap of 20.

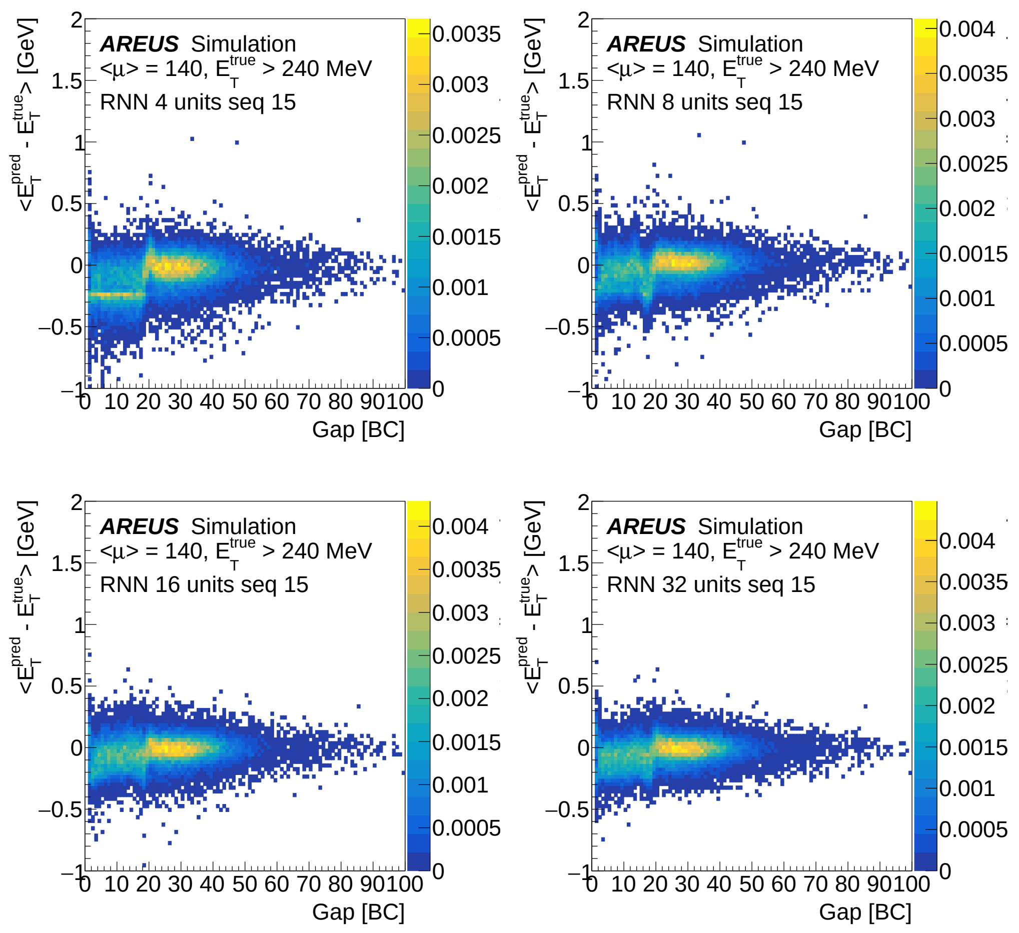


Figure 3.15: Resolution as a function of the gap between two energy deposits for simple RNNs with a sequence length of 15 and different unit sizes.

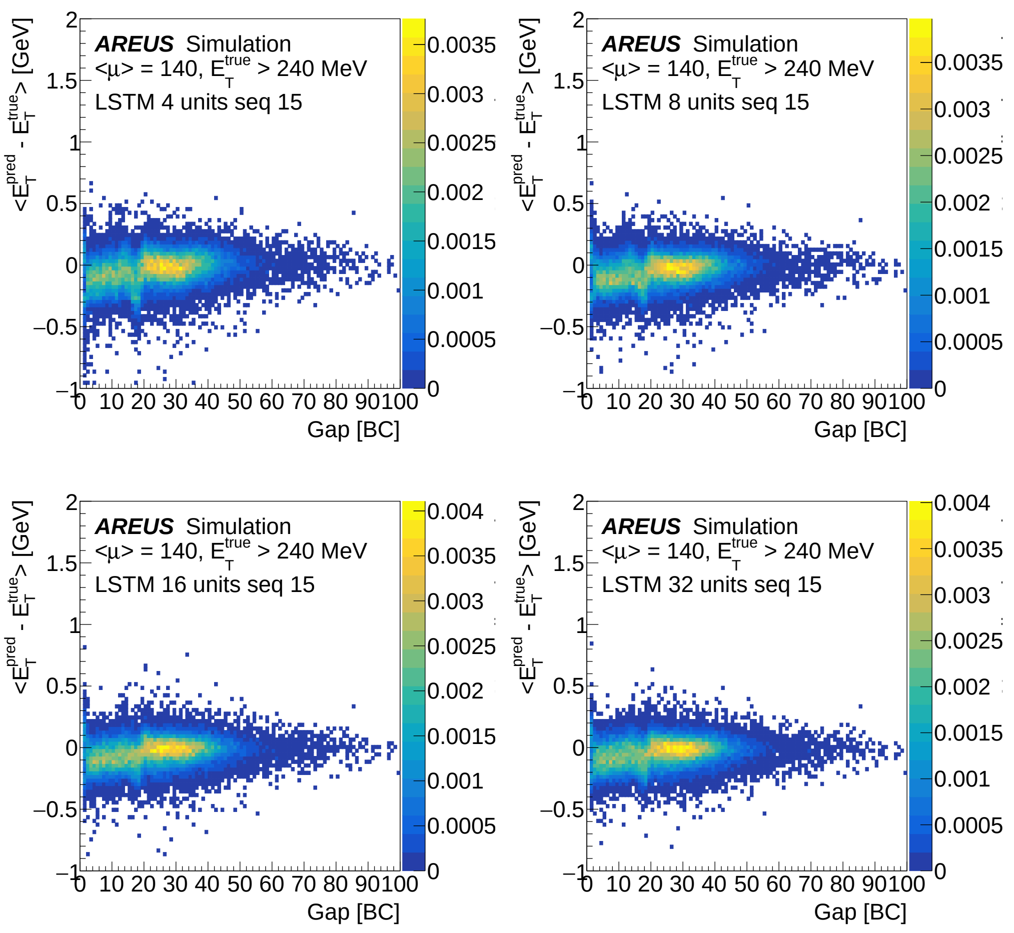


Figure 3.16: Resolution as a function of the gap between two energy deposits for LSTMs with a sequence length of 15 and different unit sizes.

Finally, using a sequence length of 30, shown in figure 3.17 for simple RNN and in figure 3.18 for LSTM, the RNNs are able to fully correct for the out-of-time pileup, in all gap regions given that the network is large enough. An excellent performance is noticed for RNNs with 16 and 32 units and LSTMs with 8, 16 and 32 units. The RNN with 4 units is performing poorly with this long sequence. The RNN with 8 units is able to correct for out-of-time pileup but the correction is not as good as the one observed with a sequence length of 15 due to the network not being complex enough to probe long sequences.

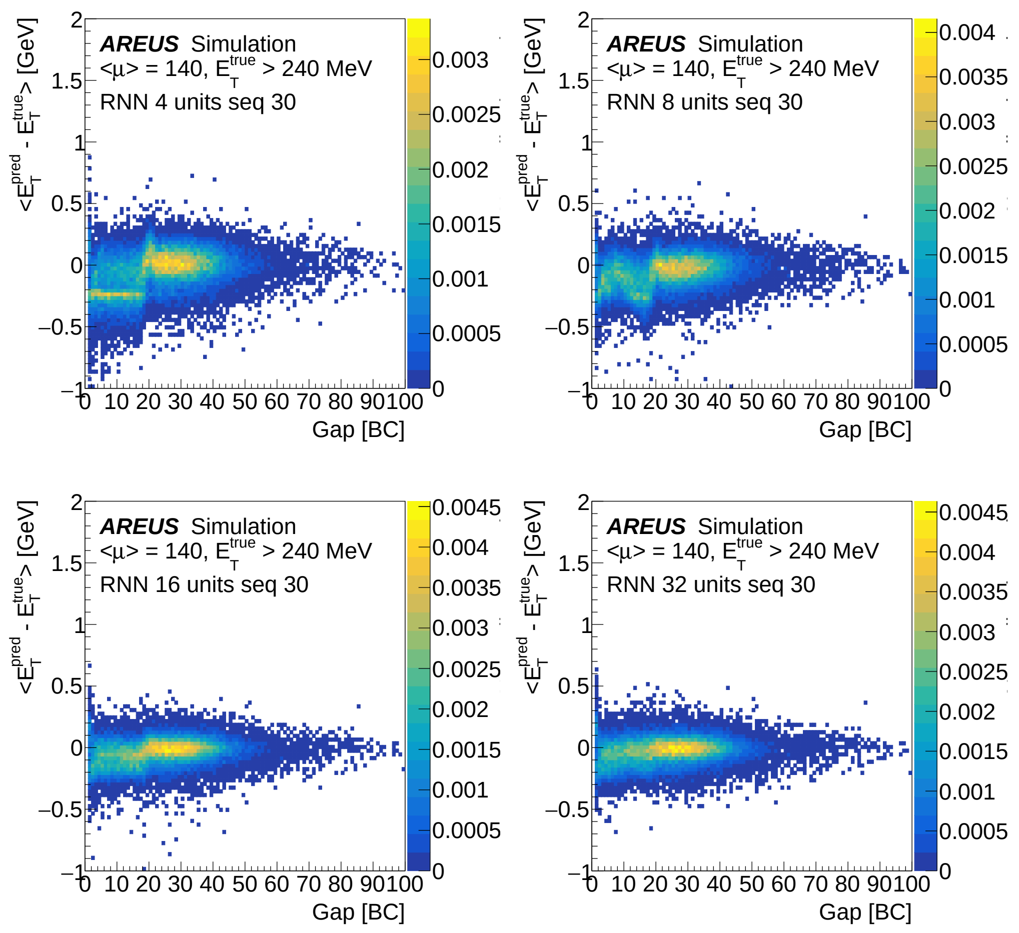


Figure 3.17: Resolution as a function of the gap between two energy deposits for simple RNNs with a sequence length of 30 and different unit sizes.

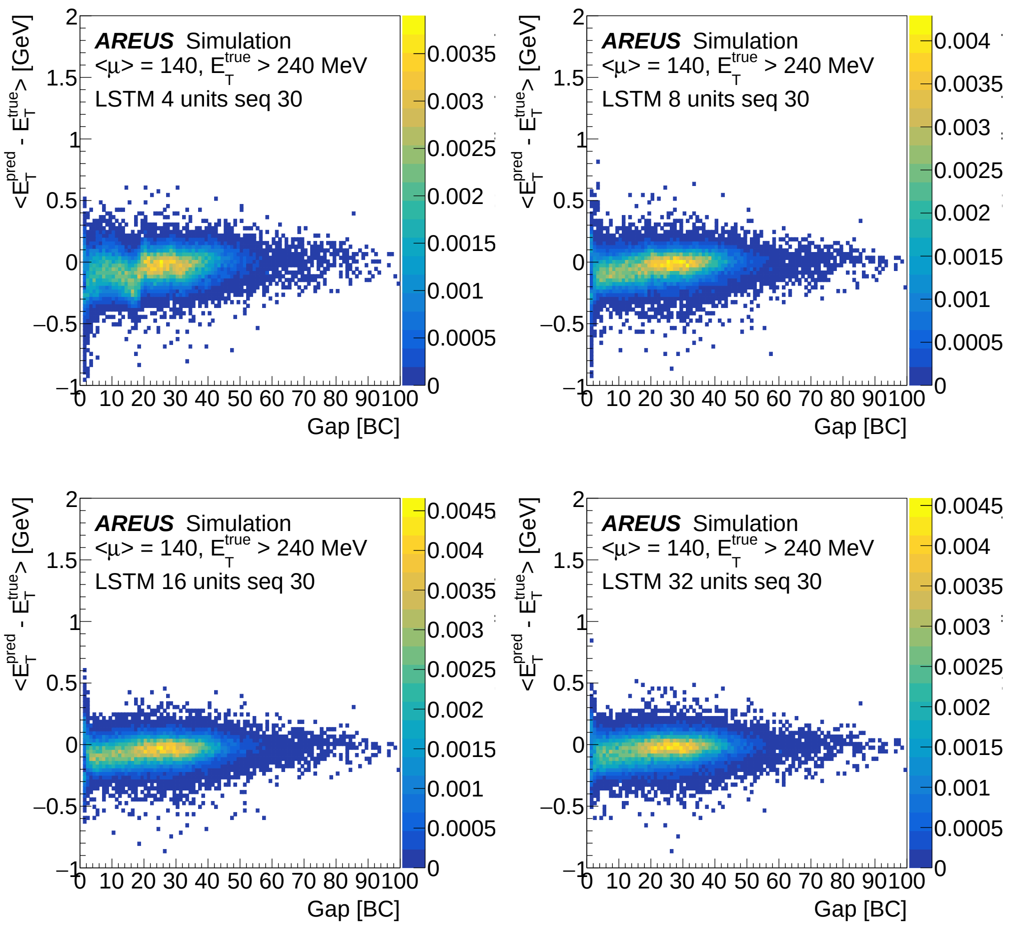


Figure 3.18: Resolution as a function of the gap between two energy deposits for LSTMs with a sequence length of 30 and different unit sizes.

The performance of the energy reconstruction methods clearly show a difference in the low gap and high gap regions. The drop of performance appears mainly when the time gap between two pulses is lower than 20 bunch crossings. It is thus useful to evaluate the performance independently for pulses at a time gap lower and higher than 20 bunch crossings.

The energy resolution for the low gap region is shown in figure 3.19 for simple RNNs and figure 3.20 for LSTMs. All networks perform slightly better than OFMax for a sequence length of 5. Adding three more samples in the past, for a total sequence length of 8, improves the performance slightly especially for large networks. The neural networks clearly perform better starting with a sequence length of 15 where the larger LSTM and simple RNNs are able to mitigate the effect of the out-of-time pileup and resolve the double peak structure. When increasing the sequence length to 30, only the simple RNNs with 16 and 32 units and LSTMs with 8 or more units are able to gain performance. The long sequence causes mainly performance decrease for the smaller networks.

Figure 3.21 and 3.22 show the energy resolution in the high gap region for RNNs and LSTMs, respectively. The sequence length has a slight effect on the performance of the networks in this region. The network size plays a more important role in this region as expected. With large network size and long sequence length the neural networks are able to achieve similar performance as OFMax in the high gap region. One should note that the optimal filtering technique is optimal for isolated pulses and the networks are not expected to outperform OFMax in this region. The purpose of the network is to correct the degraded performance in the low gap region while keeping similar performance to OFMax in the high gap region.

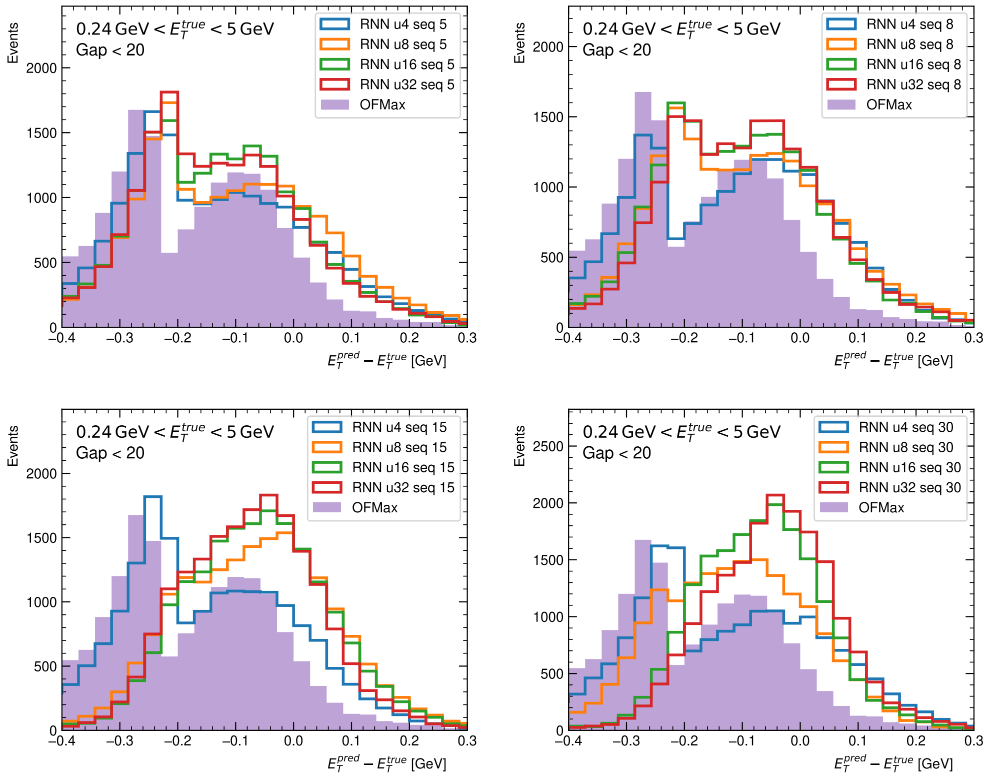


Figure 3.19: Transverse energy resolution for different simple RNNs in the region with a time gap of less than 20 bunch crossings.

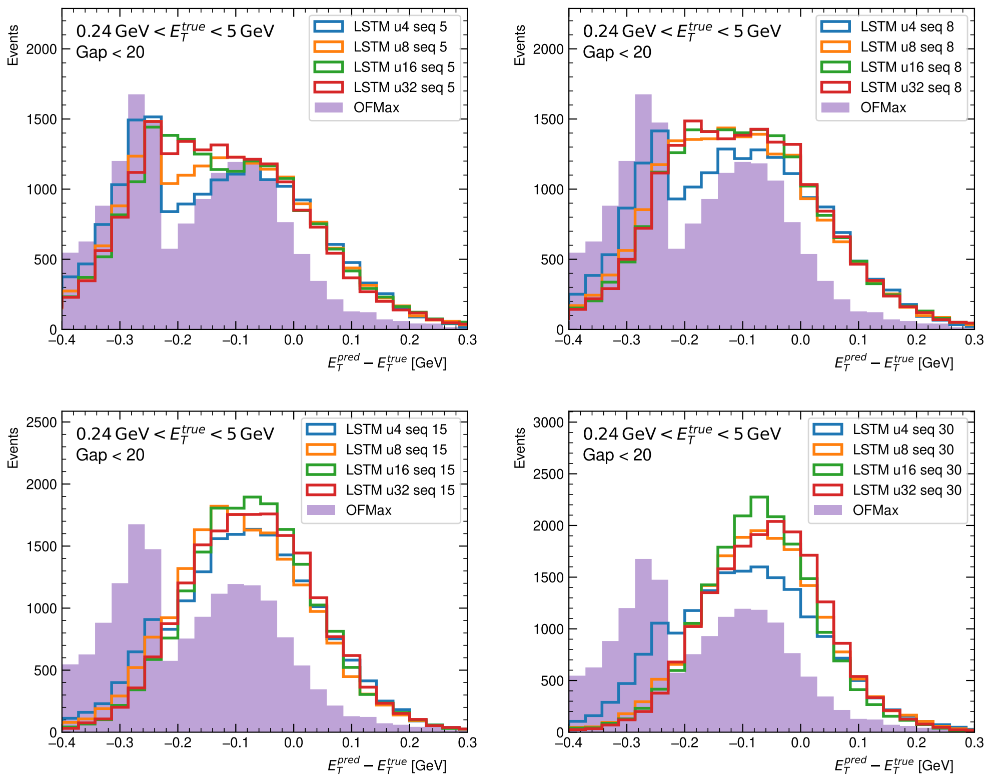


Figure 3.20: Transverse energy resolution for different LSTMs in the region with a time gap of less than 20 bunch crossings.

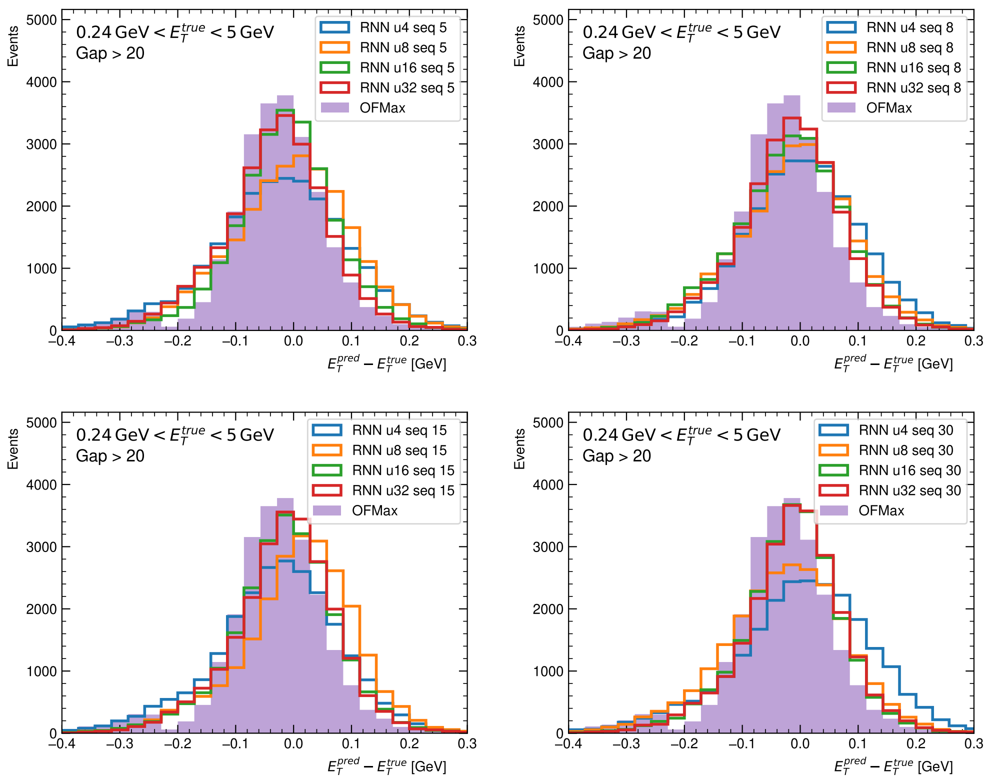


Figure 3.21: Transverse energy resolution for different simple RNNs in the region with a time gap of more than 20 bunch crossings.

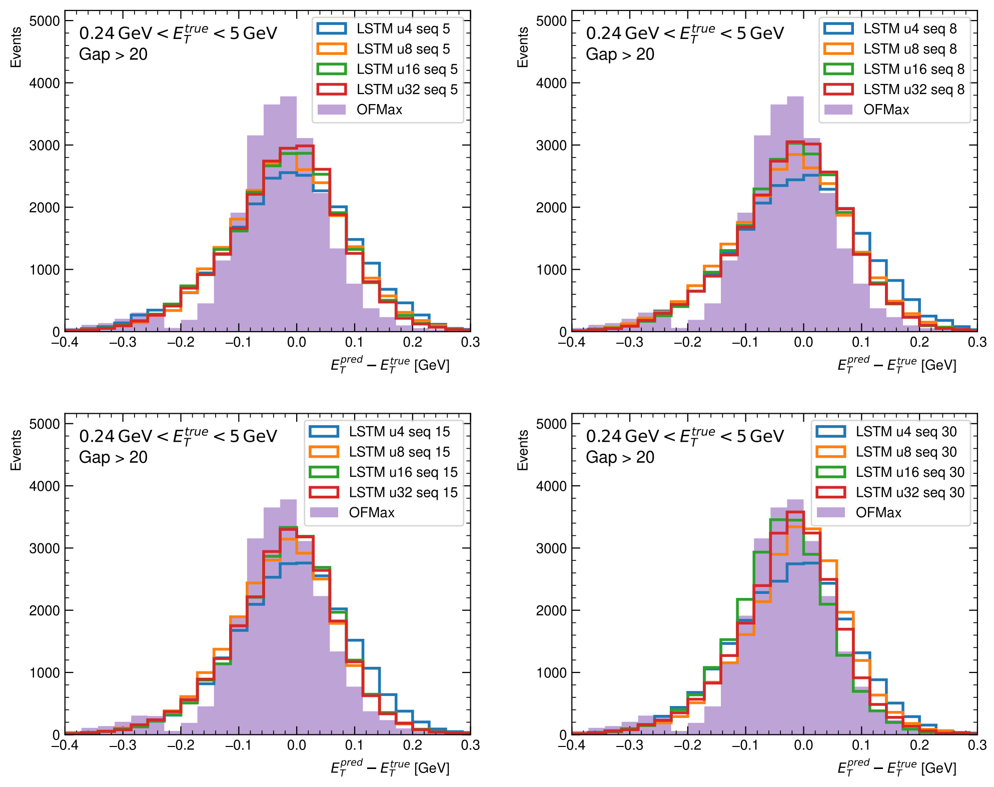


Figure 3.22: Transverse energy resolution for different LSTMs in the region with a time gap of more than 20 bunch crossings.

Figure 3.23 summarizes the network performance in the low and high gap region as a function of the sequence length and the size of the network. In the high gap region the resolution improves mainly with the increased size of the networks as expected. The neural networks show slightly better performance in this region compared to OFMax in terms of standard deviation. This is mainly due to the small second peak seen for OFMax in this region. This second peak is due to the fact that the applied cuts at gap > 20 and $E_T^{true} > 240$ MeV do not eliminate completely overlapping pulses which affects the overall performance of OFMax. In the low gap region the resolution of the neural network clearly improves with the length of the input sequence. This is however not the case when the neural network size is not large enough to carry all the information in the long sequence through the internal cell state of the network.

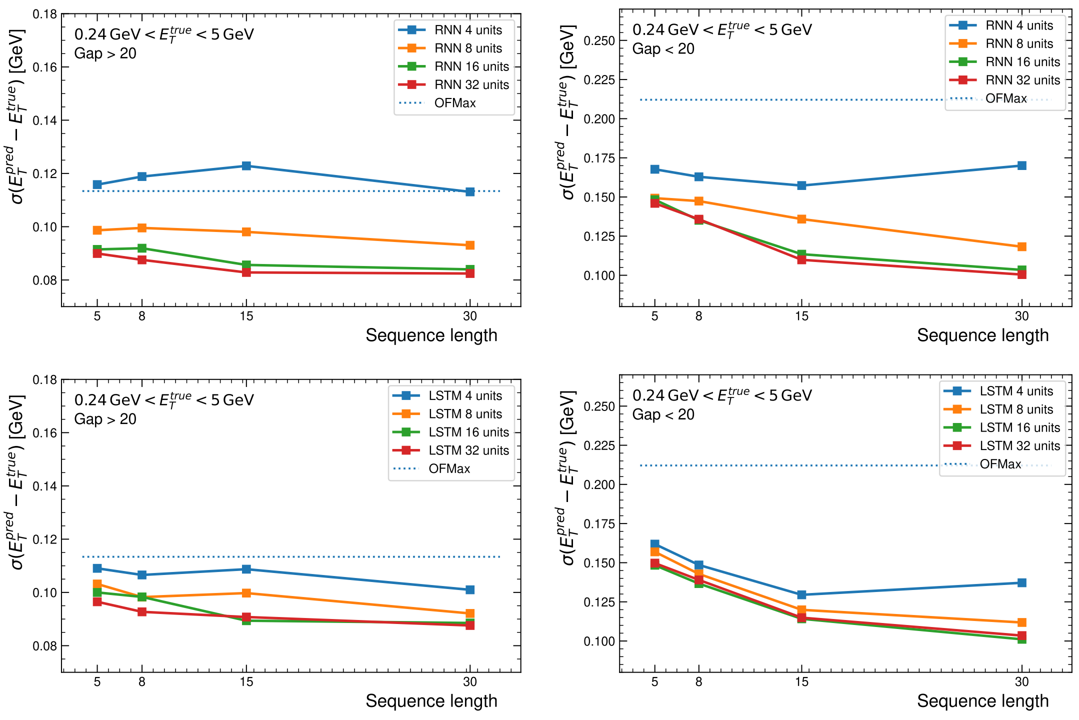


Figure 3.23: Transverse energy resolution for high gap and low region for simple RNN and LSTM.

3.4 Energy reconstruction with single-cell LSTM

In contrast to the sliding window approach, the many-to-many RNN architecture, explained in chapter 2.3, allows for energy reconstruction at each update of the RNN state. The RNN processes one input sample and updates its hidden state at each time step, without resetting the state between bunch crossings. This means that the network carries the state with full information about the past, which allows it to correct better for out-of-time pileup. Furthermore, the computation capacity needed per predicted bunch crossing is reduced in comparison to processing the full sliding window.

Figure 3.24 illustrates the processing flow of the many-to-many RNN architecture for continuous energy reconstruction. The energy is reconstructed with a delay of 5 bunch crossings which means that 5 samples around the pulse peak are used for the energy computation. A single neuron is used for the output layer to reconstruct the deposited energy. This approach allows for continuous energy reconstruction and correction for out-of-time pileup over long time ranges, making it a promising alternative to the sliding window approach. This approach is only possible with a gated RNN structure such as the LSTM.

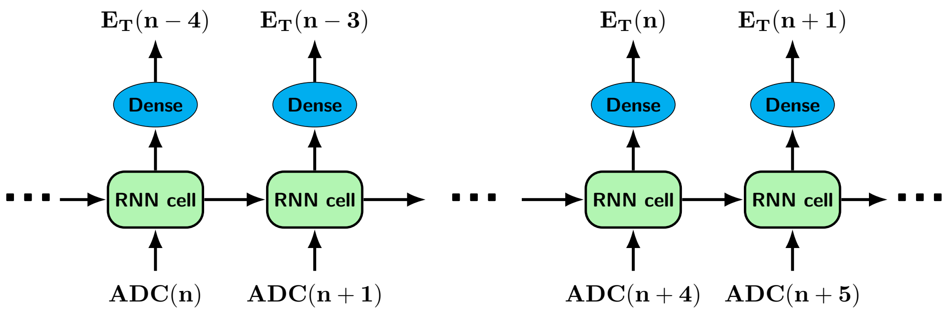


Figure 3.24: Representation of the many-to-many architecture applied for continuous energy reconstruction, where the input signal is processed by a RNN cell to reconstruct the energy [1].

3.4.0.1 Training single-cell LSTM

The single-cell LSTM is a many-to-many network where the state for each timestep is reconstructed as energy. The training dataset consists of ≈ 1M events. However training directly on such a long sequence does not converge. To train a single-cell it is mandatory to reset the states in the initial part of the training after a limited sequence of data because initially the LSTM is not able to keep the state stable over longer sequences. After the network has learned to reconstruct short sequences, the training can be extended to long sequences without state resets. The full training procedure is explained in pseudo code block 1.

Algorithm 2 The training procedure for single-cell LSTM by training it with incrementally longer sequences.

```
1:  for $sequence$ in $[50,100,200,250,300,400,500,700,1000,1500,2000]$ do
2:      for $m = 1,2,\ldots,20$ do
3:          Select random starting point $n$ in the training set
4:          Fit the model with $sequence \times 100$ continuous bunch crossings starting at
            $n$ with weight updates every $sequences$ bunch crossings
5:          Reset the LSTM states to a vector of zeros
6:      end for
7:  end for
8:  for $sequence$ in $[200,500]$ do
9:      Fit the model for 200 epochs with weight updates every $sequence$ bunch crossings
10: end for
11: Fit the model for 5000 epochs
```

3.4.0.2 Performance

Figure 3.25 shows the performance of single-cell LSTM in comparison to sliding window LSTM with a sequence length of 5. Both of the networks have 10 units that amounts to 491 parameters for the full network. The single-cell LSTM is able to better correct for overlapping events than the sliding window with sequence length of 5. Figure 3.26 shows the resolution of the single-cell LSTM in comparison to sliding window LSTM and simple RNN. The overall performance of the single-cell LSTM is better than with the sliding window methods. Figure 3.27 shows the energy resolution in the high gap and low gap regions.The single-cell LSTM has better performance in the low gap region as expected. However this comes at the price of a degradation of the performance in the high gap region. All networks perform better than OFMax in the low gap regions. In the high gap region only the sliding window RNNs outperform OFMax.

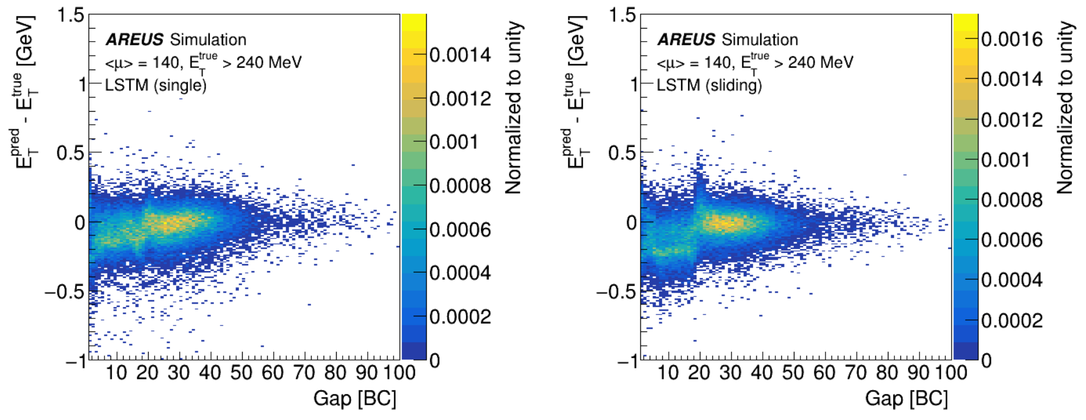


Figure 3.25: Transverse energy resolution as a function of the time gap for (left) single cell LSTM and (right) sliding window LSTM [1].

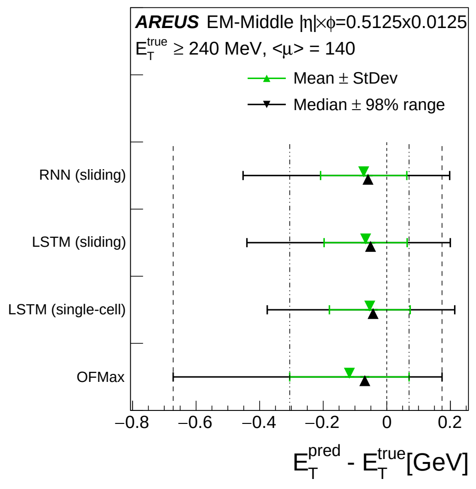


Figure 3.26: Transverse energy resolution for a single-cell LSTM, sliding window LSTM with a sequence length of 5, and a simple RNN with a sequence length of 5 compared to OFMax. The standard deviation and the 98% range are shown [1].

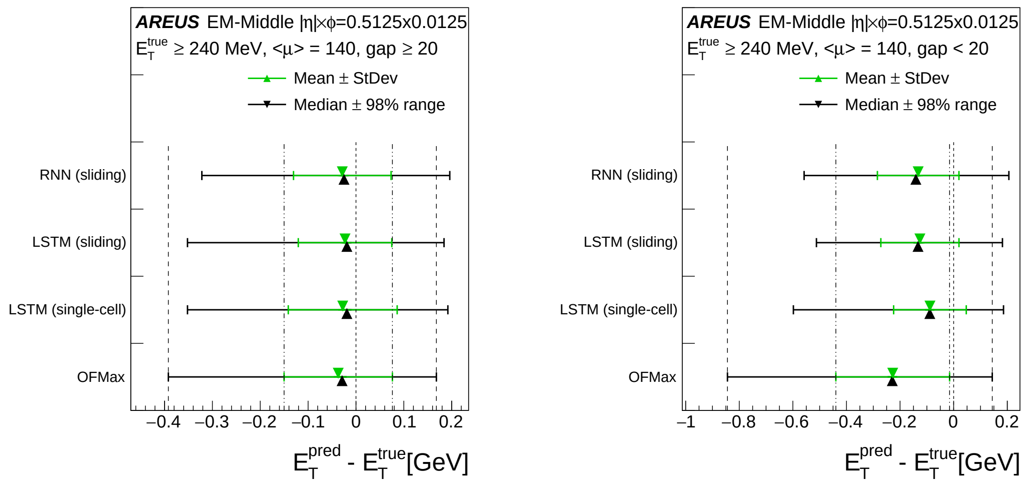


Figure 3.27: Transverse energy resolution for a single-cell LSTM, sliding window LSTM with a sequence length of 5, and a simple RNN with a sequence length of 5 compared to OFMax. The standard deviation and the 98% range are shown for (right) high gap and (left) low gap [1].

3.4.0.3 Feasibility

The single-cell LSTM faces several challenges that limit its feasibility for the real-time reconstruction. One key challenge is that the network requires a stabilization period when starting from the zero state, which limits its performance before the state stabilizes. Another related issue are unexpected detector effects, such as noise bursts. These detector effects are difficult to accurately simulate when generating the training data. The effects from noise bursts are limited for the sliding window algorithms since the state is reset at each bunch crossing. However the single-cell LSTM will carry the information about the noise bursts over the full run period. This can result in unexpected behavior over long period of time while the system state stabilizes. Another challenge for single-cell LSTMs is the need to simulate long sequences for the training. This is not possible in the current ATLAS framework that simulates the full detector. It is only possible for single calorimeter cells that do not take into account the full detector geometry. Implementing the simulation of long sequences for the full detector is currently out of reach since it will require a huge amount of CPU and memory. As a result, the performance of the detector simulation may not match the performance observed during data taking. Furthermore problems in the firmware implementation arise since the single-cell architecture, the computation at BCID n needs to wait for the results of the computation at BCID $n-1$. In this case the computation at each BCID should not take more than 25 ns. This is hard to achieve as it was shown in [1].

For these reasons, while the single-cell LSTM is able to perform well and has excellent correction for out-of-time pileup, it is not a viable candidate for the energy reconstruction in the LASP board.

3.5 Timing reconstruction

In addition to energy reconstruction, the OF algorithm also computes the time shift, also referred to as the phase shift, of the pulse. The OF algorithm actually computes the product of the pulse amplitude times that phase shift as seen in equation 1.5. This value can be used to determine the BCID of the deposited energy and also to detect the presence of long-lived particles [22].

The computation of the phase shift can be included in the RNNs with an additional output node for the network. In this case the RNN state is connected to two neurons: one that outputs the energy and one the phase shift. This allows the network to simultaneously predict both the energy reconstruction and the phase shift of the pulse. The difference to the OF computation is that the network directly outputs the phase shift value instead of the product of the energy and the phase shift.

3.5.1 Dataset with phase shift

The dataset used for this study is similar to the one used for the energy computation described in section 3.2 with the exception of having phase shift ranging from -8ns to 8ns. However AERUS is able to apply the phase shift only on signal energy deposits and not to the pileup deposits. Thus only BCIDs with signal energy deposits are selected for the training. At these BCIDs, the energy deposits coming from in-time pileup do not have a phase shift but they are added to the signal energy deposits. The phase shift of the sum is assumed to be the one from the signal energy deposit since this energy is much larger than pileup energies.

Figure 3.28 shows how the sampled pulse shape changes with the phase shift. The main changes are in the amplitude of the samples before and after with maximum amplitude.

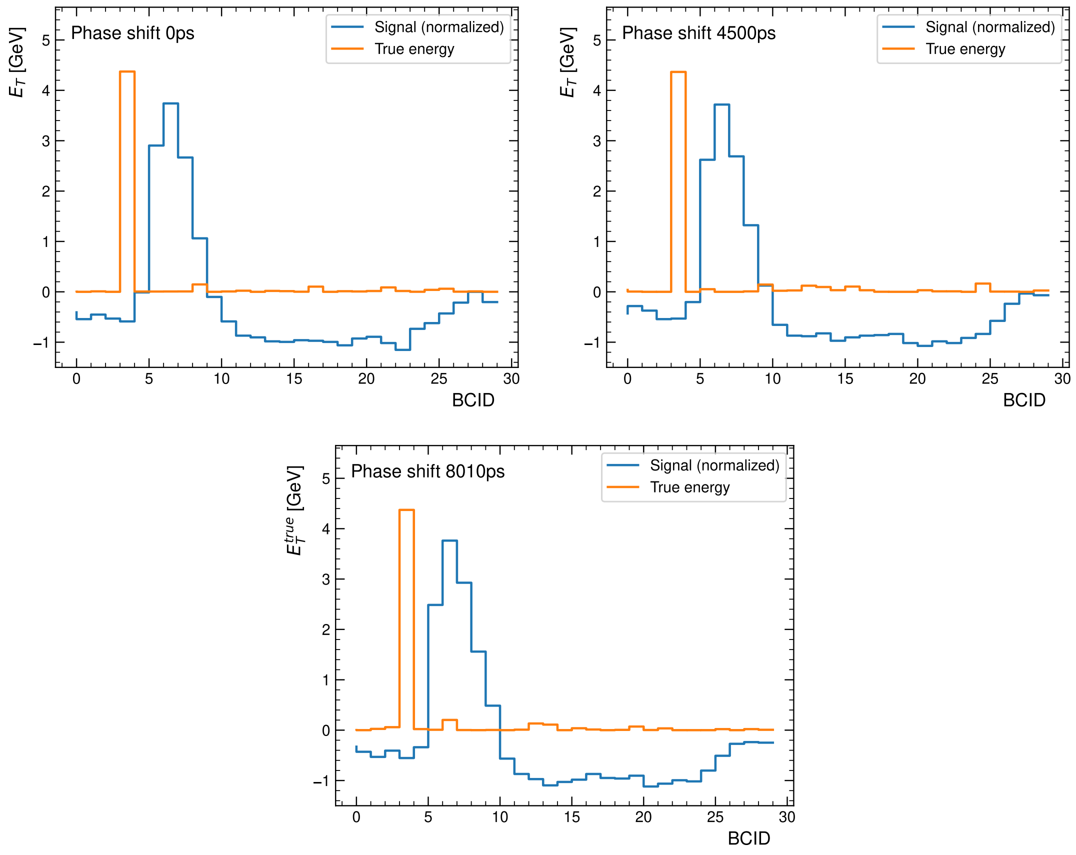


Figure 3.28: Example sampled pulse shape for different phase shift values.

3.5.2 Timing reconstruction performance

The performance is evaluated for a simple RNN with 8 units and a LSTM with 10 units both with a sequence length of 5. Figure 3.29 shows the timing resolution as a function of the true energy for networks predicting both the phase shift and the energy in comparison to optimal filtering. Both networks as well as OF show similar behavior where the phase shift is well reconstructed at high energies while the timing resolution degrades at low energies. This is expected as the effect of noise and pileup causes proportionally larger distortion to the pulses. Both LSTM and simple RNN have better performance for lower energies than OF. Figure 3.30 shows the timing resolution for networks predicting both timing and energy. Both neural networks have better resolution compared to the OF algorithm.

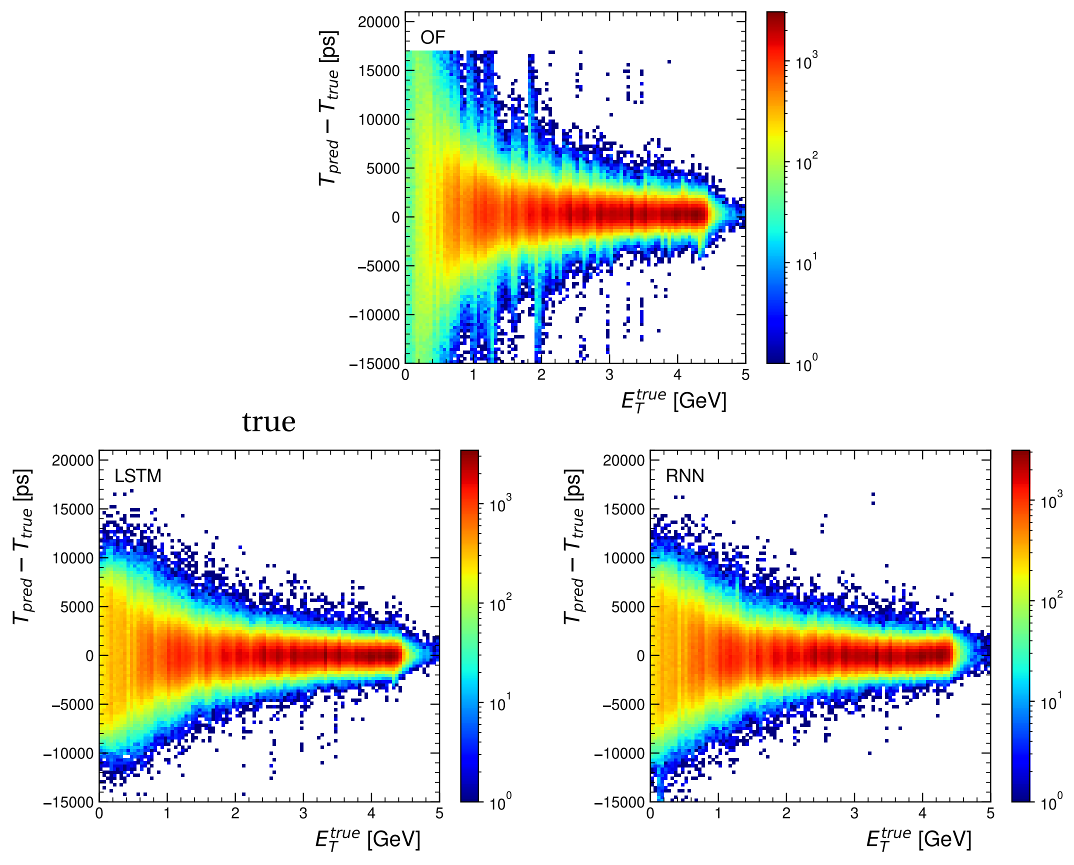


Figure 3.29: Phase shift resolution as a function of the deposited energy for the OF algorithm, the LSTM and the RNN.

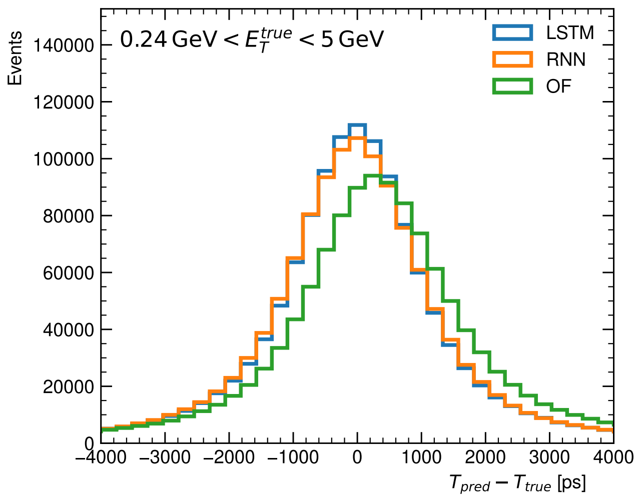


Figure 3.30: Phase shift resolution resolution $T_{pred} - T_{true}$ for LSTM, simple RNN and OF.

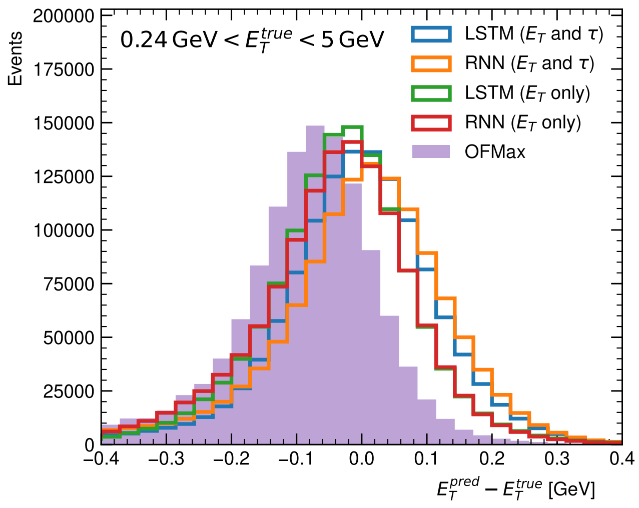


Figure 3.31: Resolution of the transverse energy for networks with and without timing predictions in comparison to OFMax.

In the network design the same RNN state is used to reconstruct both the energy and the phase shift with only one additional neuron to compute the time shift. This allows for the probing of the phase shift with minimal additional computational power. However this introduces a correlation between the resolution on the energy and on the time. Figure 3.31 shows the energy resolution for networks with and without timing predictions. One can notice a notable drop in the energy resolution when the network is required to compute both the energy and the time. A mitigation for this drop is presented in section 3.5.3.

3.5.3 Loss function to prioritize timing or energy

The MSE loss function is used to train the models, which computes the element-wise squared error and averages over these values. However, this loss function treats the energy and timing predictions equally and does not take into account the relative importance of each quantity. The value of this loss is dependent on the magnitude of the values which in turn is determined by the normalization of each of them. To address this, it is possible to use a custom loss function based on the MSE that computes the loss for the energy and the timing separately and balances the contribution of each using an additional parameter that can be adjusted accordingly to give the desirable performance. The loss function is shown in equation 3.3 where w is the weight factor given to the timing predictions.

$$L = MSE(e) + w \cdot MSE(t) \tag{3.3}$$

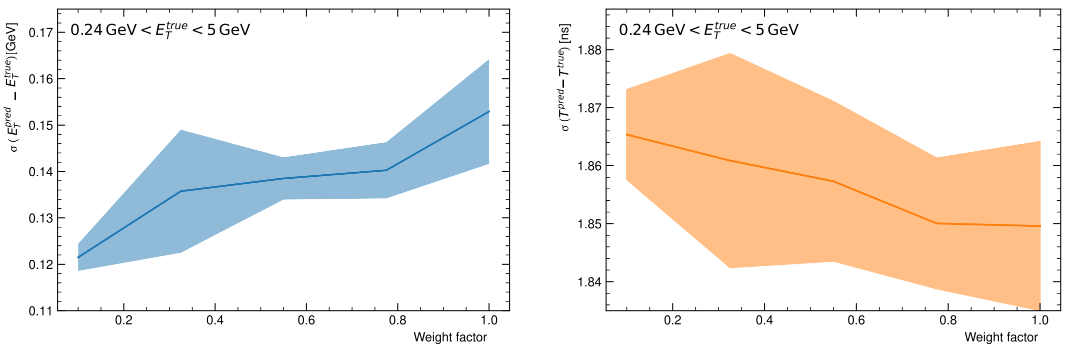


Figure 3.32: Resolution of the transverse energy and of the phase shift as a function of the weight factor that balances the corresponding contributions in the loss function for a 8 unit simple RNN.

Figure 3.32 shows the performance of 8 unit simple RNN trained with different weight factors w demonstrating that when the weight is chosen to prioritize energy, the energy resolution is improved going from a standard deviation of 120 MeV to 180 MeV, while the timing prediction does not change significantly going from 1.86 ns to 1.85 ns. This suggests that the model is able to learn to prioritize energy reconstruction without sacrificing timing resolution. Conversely, when the importance of timing is high with larger weight factor w, the timing resolution is slightly improved, but at the expense of energy reconstruction. Therefore, by using the custom loss function with a weight value that puts a higher priority on the energy resolution, it is possible to achieve a good compromise between energy and timing resolution.

Figure 3.33 shows the distribution of energy and timing resolution for different weight values. The changes in the energy resolution are significant and only the RNN with the lowest weight value of $w = 0.1$ surpasses the performance of OFMax. However the timing reconstruction performance for all weight values are better than OF and have only minor changes between each other.

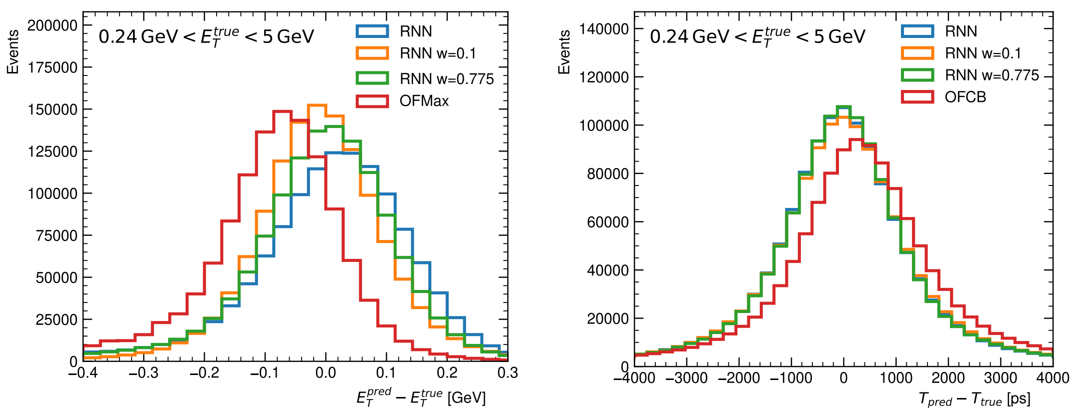


Figure 3.33: Transverse energy and timing resolution for an RNN trained with different weight factors in the loss function compared to the resolution given by OF algorithms.

3.6 Resilience against changing instantaneous luminosity

The nominal luminosity range for the HL-LHC is expected to be $5-7\times10^{34}\text{cm}^{-2}\text{s}^{-1}$ which translates to 140 to 200 proton-proton interactions per bunch crossing. The pileup rate changes between LHC runs and also during the run on a BCID-by-BCID basis. Because of this the RNNs need to be resilient against possible changes in the luminosity. The optimal filter coefficients are acquired for each expected average pileup rate independently and are known to not perform optimally over varying pileup with higher average pileup rates. This section evaluates the ability of simple RNNs to cope against varying pileup rates. The selected network configuration is the 8 unit simple RNN with a sequence length of 5.

3.6.1 Performance of the RNNs on samples with varying average pileup rate

Three datasets with an average pileup rate of 100, 140 and 200 are generated using AREUS. Since the RNN performance depends on the initialization of the weights at the training level, 100 RNNs are trained for each pileup rate with a different random initialization. This allows for the separation of the effect of the changing pileup from the initialization effects. An additional 100 networks are trained with an equal mixture of the sample with pileup 100, 140, and 200. In total 276 models converged and their performance is evaluated on the three datasets with different pileup.

Figure 3.34 shows $\sigma(E_T^{pred} - E_T^{true})$ for each or the RNNs trained and applied to the different samples with different $\langle\mu\rangle$, each model being an entry in the histogram. The resulting distributions have large overlap between the RNNs trained with different pileup conditions which suggests that, overall, the impact of the initialization of the model has a larger effect on the performance than the pileup conditions used in the training dataset. This indicates that the way in which the model is initialized can have a significant impact on its ability to learn and generalize to unseen luminosities. While the overlap of the histograms is significant, there is a trend where the models trained with higher pileup are more likely to converge to better energy resolution.

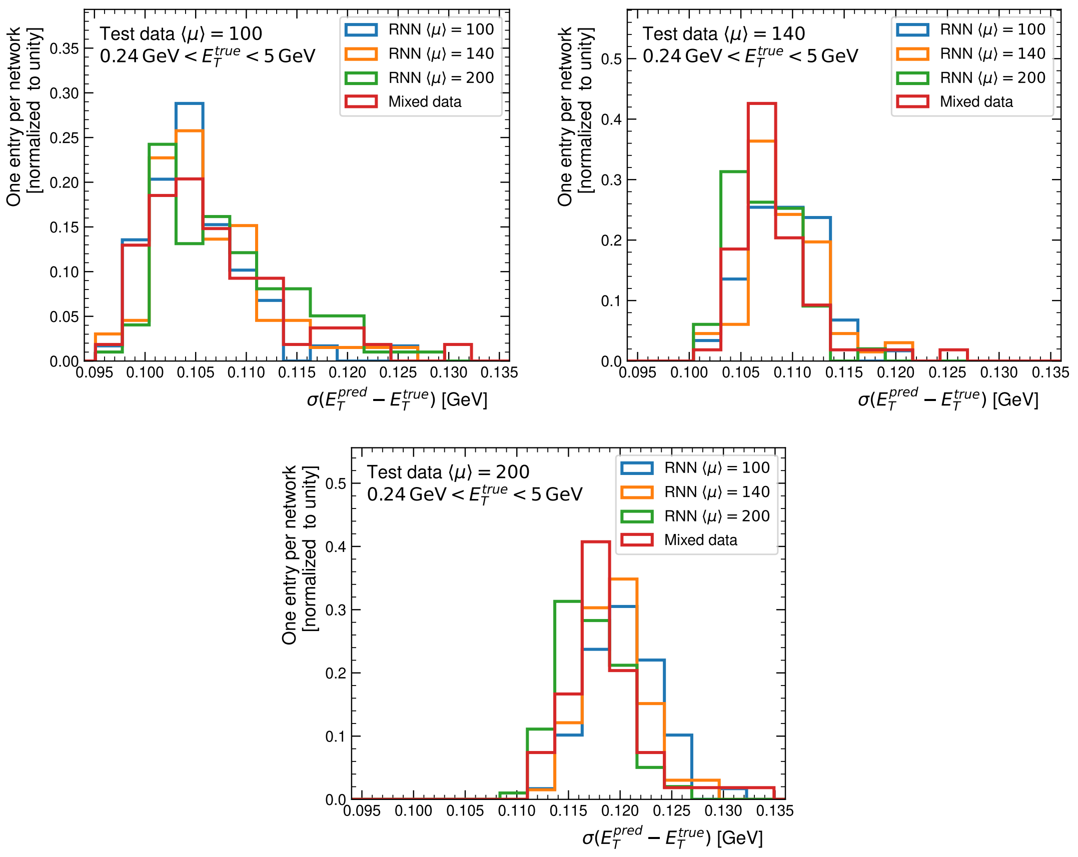


Figure 3.34: Distribution of $\sigma(E_T^{true} - E_T^{pred})$ obtained by changing the initialization of the RNN for different values of $\langle\mu\rangle$ in the training sample. The RNNs are applied to (upper left) a dataset with $\langle\mu\rangle = 100$, (upper right) a dataset with $\langle\mu\rangle = 140$ and (bottom) a dataset with $\langle\mu\rangle = 200$. Only events with $E_T^{true} \geq 240$ MeV are considered when evaluating the performance of the networks [61].

3.6.2 RNN resilience against pile-up

To evaluate the performance of the RNNs as a function of the $\langle\mu\rangle$ values, the model with the best resolution is chosen from each set of models trained on a given pileup sample. By comparing the performance of the best models trained with different pileup rates, we can gain insights into how well they can generalize to real data taking conditions, where pileup rates can vary widely. The performance of the RNNs are compared to OFMax with coefficients extracted with a pileup ranging from $\langle\mu\rangle = 100$ to $\langle\mu\rangle = 200$ with steps of 20.

Figure 3.35 shows the performance of the networks trained with different $\langle\mu\rangle$ in comparison to OF with different coefficients as a function of the $\langle\mu\rangle$ value. The resolution of all models degrade at high pileup as expected. However all RNNs have an energy resolution of about 50 MeV better than OFMax. It is also notable that for the RNNs the lowest standard deviation is achieved by the model trained with a pileup rate of 200 and the lowest performing network was trained on pileup of 140. This could be due to the initialization during the training phase that can affect the performance of the neural networks.

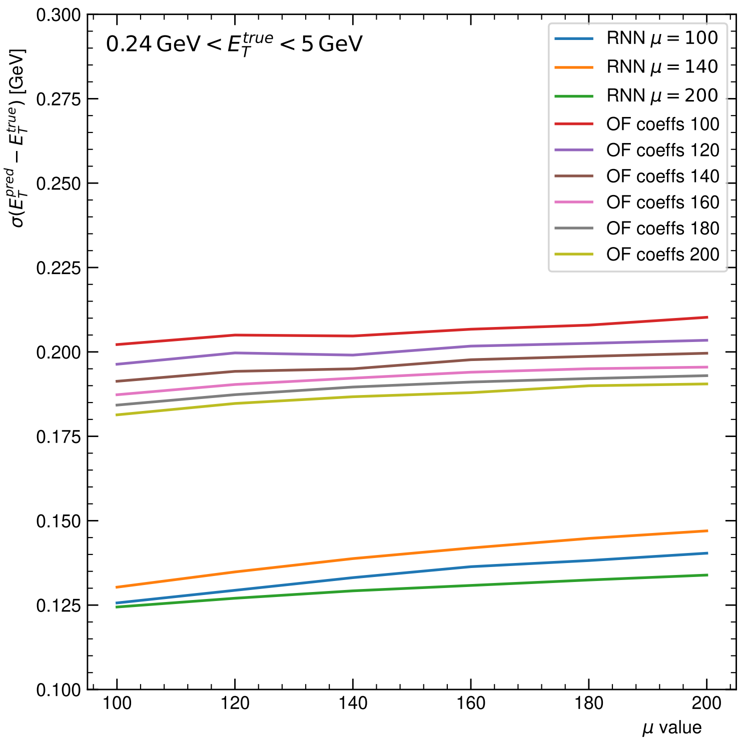
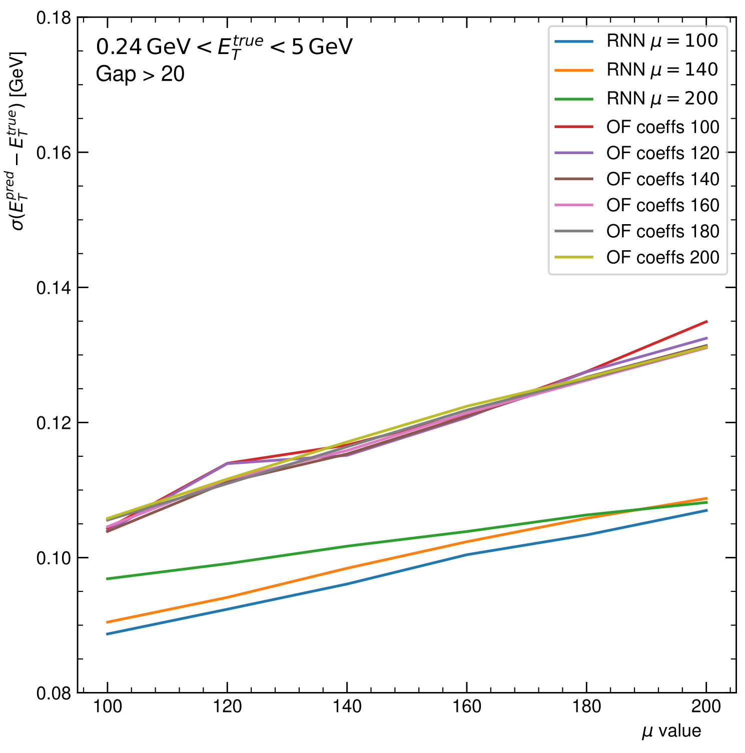


Figure 3.35: $\sigma(E_T^{true} - E_T^{pred})$ as a function of the $\langle\mu\rangle$ value for various RNNs and OFMax trained with different $\langle\mu\rangle$ values.



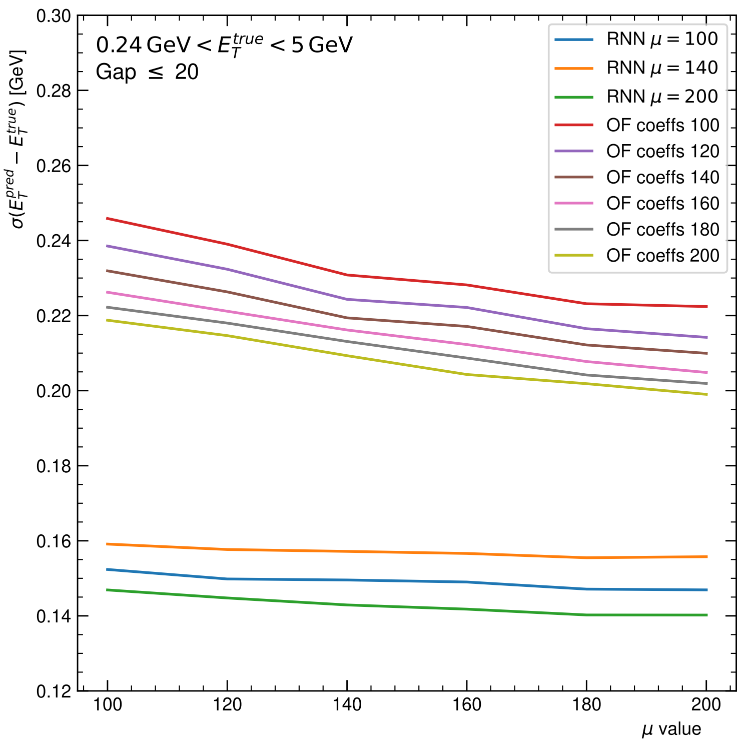


Figure 3.36: $\sigma(E_T^{true} - E_T^{pred})$ as a function of the $\langle\mu\rangle$ value for various RNNs and OF-Max trained with different $\langle\mu\rangle$ values for (top) high time gap between two pulses and (bottom) low time gap between two pulses.

Figure 3.36 shows the neural networks and the OFMax resolution as a function of $\langle\mu\rangle$ in the high and low time gap regions separately. In the region with a high gap between two pulses the performance decreases when the pileup increases as expected since both RNNs and OFMax are further affected by the noise from pileup. One should note that the time gap is computed for deposits with $E_T^{true} > 240$ MeV so even if pulses at high gaps are relatively well isolated they are still affected by noise from low energy deposits. The RNN trained on $\langle\mu\rangle = 200$ has smaller drop in performance than the two other networks, however the best performing network is trained with $\langle\mu\rangle = 100$. In the low time gap region the networks have almost invariant performance where increased pileup noise does not decrease the performance while the standard deviation of OFMax slightly decreases when the pileup increases. This effect is not fully understood and points to the complex balance between reconstructing the energy in isolated pulses and correcting for overlapping pulses that these algorithms exhibit.

3.7 Conclusion

In this chapter the viability of two different RNN types was evaluated for the reconstruction of the energy deposited in the LAr calorimeter. In the sliding window configuration both the simple RNN and LSTM are outperforming OFMax given a sufficiently large network. The improvements are in large part due to the ability of the RNNs to correct for out-of-time pileup. Networks with 16 units or more are able also to match the OFMax performance for isolated pulses.

The single-cell architecture with an LSTM cell is found to perform best for overlapping pulses. However this architecture is not optimal for isolated pulses. This architecture requires the simulation of very long sequences which is not feasible for all cells in the calorimeter. Moreover this architecture does not fit the latency requirements to be implemented in the LASP FPGAs. These two limitations make this architecture not viable in the LAr energy reconstruction.

The RNNs are expanded to predict possible timing phase shift of the pulse by adding a second output to the neural network. The timing resolution of both simple RNN and LSTM is better than the OF algorithm. A custom loss function is built allowing to prioritize the performance of the RNN in terms of energy resolution or timing resolution during the training. Prioritizing energy resolution improves the energy resolution while having negligible effect on the time resolution.

The instantaneous luminosity is expected to change during HL-LHC runs. Therefore the RNNs should be resilient against changes in the average pileup since it is not possible to change the neural network weights during the run. RNNs trained and applied to different $\langle\mu\rangle$ values have shown resilience against pileup changes. RNNs trained on samples with pileup ranging from $\langle\mu\rangle = 100$ to $\langle\mu\rangle = 200$ can be applied to data with pileup in the same range without significant loss of performance.

4 Optimizing Neural Networks for FPGAs

Summary

4.1 Introduction

The real-time processing of LAr calorimeter data is done by custom electronics boards based on FPGA chips. FPGAs are reprogrammable which makes them highly versatile. They consist of a matrix of programmable logic blocks (PLBs) that can be configured to perform different logic functions (AND, OR and XOR). In addition to the simple logic functions, FPGAs contain blocks to enable more complex operations, these include digital signal processing blocks (DSPs) which implement multiplication and division as well as the random access memory (RAM). The PLBs can be connected through programmable interconnects to create the desired custom circuit. They allow the creation of systems that repeatedly perform the same computations with high throughput and low latency which makes them the best candidates for the online energy reconstruction in the LAr calorimeter.

Deploying machine learning models on FPGAs has become increasingly popular due to their flexibility, parallel processing capabilities and energy efficiency. FPGAs give much of the benefits of using application specific integrated circuits (ASICs) but give a high level of configurability. However the deployment of these models pose challenges due to the nature of FPGAs. While modern FPGAs implement floating point operations, which is what machine learning models are usually trained with, these operations consume a large amount of resources. Instead the preferred way to perform computations in FPGAs is using fixed-point arithmetic, which is a method to represent fractional numbers. Fixed point arithmetic allows for faster computation with lower resource utilization, however they provide a relatively smaller dynamic range than floating points and thus have lower precision. The fixed point representation of numbers is defined by its total number of bits W and the number of integer bits I. The ranges of unsigned fixed-point is shown in equation 4.1 and of signed fixed-point in equation 4.2 and the quanta for both is 2^{I-W} [62]. This reduced numerical precision requires special care for the implementation of the calculations or else the accuracy of the neural network is significantly reduced due to the quantization.

$$x \in [0, (1-2^{-W})2^{I}] \tag{4.1}$$

$$x \in [-0.5 \cdot 2^{I}, (0.5-2^{-W})2^{I}] \tag{4.2}$$

The firmware implementation of the RNNs developed in chapter 3 has been already performed and the results are published in [1] and [2]. These implementations used neural networks trained with floating points and implemented using fixed points in firmware. The conversion from floating point to fixed point lead to loss of accuracy however the number of bits in the fixed-point representation was increased to reduce this effect. In this chapter quantization aware training will be investigated to perform the training of the neural networks with fixed-point arithmetic. This allows to further reduce the number of needed bits, and thus the resources in the FPGAs, without loss of accuracy. Other methods to reduce the resources such as pruning the networks and a NN-based initialization of the RNNs are also investigated.

4.1.1 DSP arrangement in Agilex FPGAs

The chosen FPGA type for the LASP board is the Intel Agilex FPGA. The previous implementation [2] of the RNNs with firmware targeted the INTEL Stratix 10 FPGA since the Agilex was not available. Each Stratix 10 DSP allows to perform two fixed-point multiplications with 19x18 bits with a single DSP block. Thus the implementation of the RNN can use up to 18 bits to represent the weights and the data. However each Agilex DSP allows to perform four 9x9 bits multiplications with a single DSP block [63]. Figure 4.1 shows the Agilex DSP arrangement in the 9x9 mode. To take full advantage of this new DSP arrangement and to reduce the number of needed DSPs by a factor of two, the neural networks running in the LASP should be able to run accurately with 9 bit fixed-points operations.

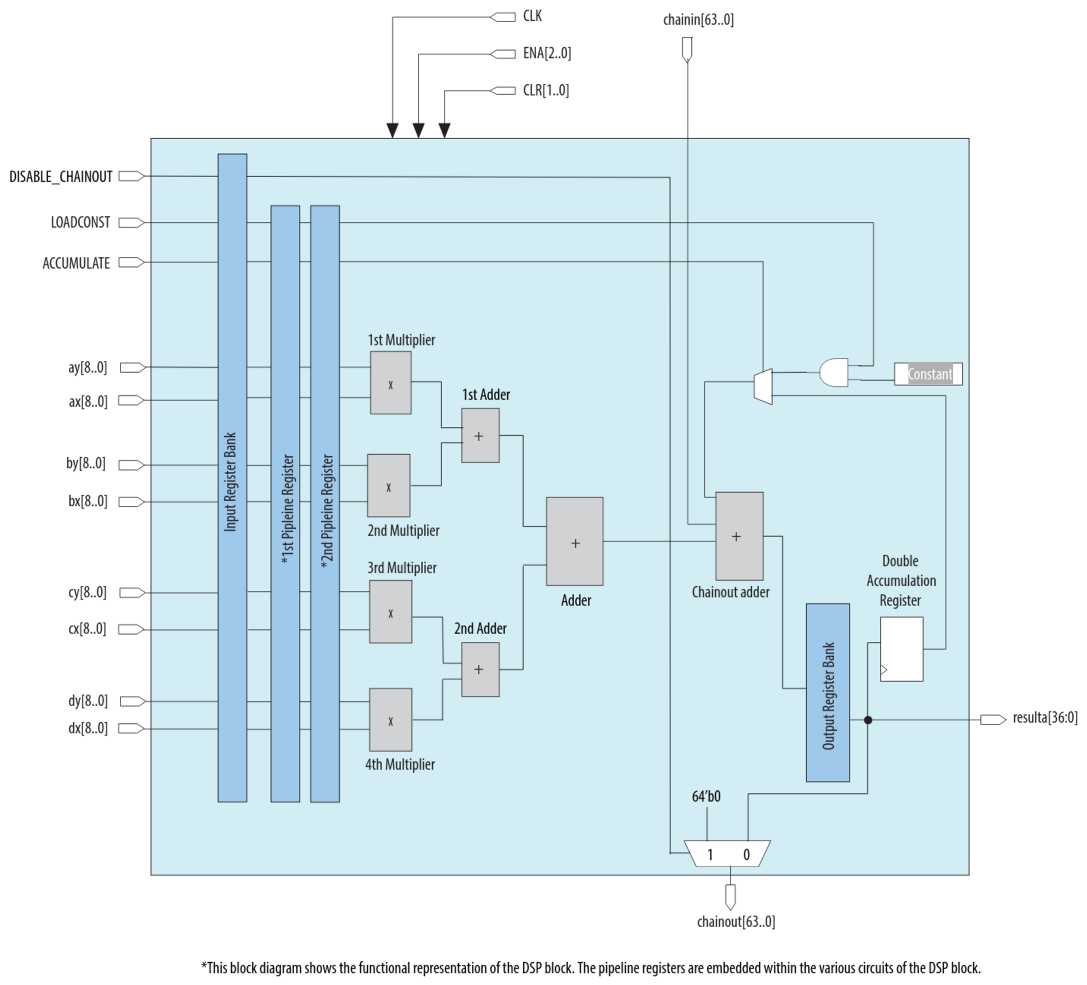


Figure 4.1: Agilex DSP operated in a mode where four 9x9 bits fixed-point operations are allowed [63].

4.2 Quantization Aware Training

Quantization Aware Training (QAT) [64] is a method of training neural networks which mitigates during training time the loss in performance caused by using lower bitwidth numerical representations, commonly known as quantization error. This method reduces the quantization error in comparison to post-training quantization (PTQ) where the appropriate bitwidth is determined after training the neural network in floating point precision. When doing PTQ, the FPGA implementation needs sufficient bitwidth to replicate the operations without significant loss in performance which usually requires high bitwidths of around 16 bits. Quantization after training also poses the problem for the energy reconstruction in LAr due to the fact that the reconstruction in full detector requires several sets of weights that are used for different calorimeter cells. This has the risk of requiring possible changes in the FPGA implementation when new weights are needed due to changes in the calibrations in case the previously determined bitwidth is not sufficient. This problem can be avoided by QAT in which the bitwidth is determined before the training and the network learns to perform the operations with the pre-determined bitwidth.

Quantization aware training can be done by using a quantization function shown in equation 4.3 where W is the total bitwidth, I is the bits used for the integer part, x is the input [3]. The problem of training neural networks with quantized numeric representations is that by using a quantization function the gradients are equal to zero due to the function being a step-like with a step size of the quanta. Zero gradients prevent the usage of backpropagation. The straight-through estimator (STE) [65] can be used during the backwards pass to enable the usage of backpropagation for the training. The STE is a linear estimation of the quantized function which is used for the gradients. In the forward pass a quantization function is used to turn the weights and activations to fixed-point values and in the backward pass the gradients are estimated using the STE. Figure 4.2 shows the quantizations of weights and activation and their straight-through estimators. This way of training the networks allows for the mitigation of the quantization error during training time.

$$y = 2^{I-W+1} \cdot clip(round(x \cdot 2^{W-I-1}), -2^{W-1}, 2^{W-1} - 1) \tag{4.3}$$

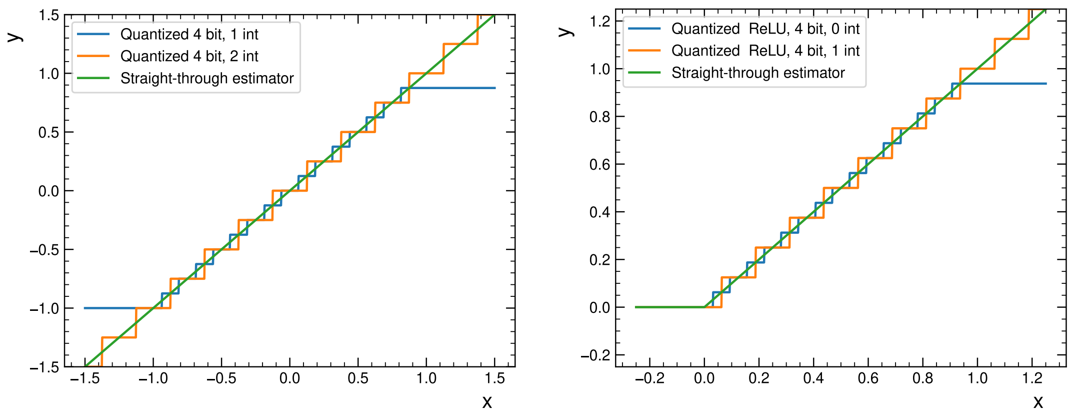


Figure 4.2: Quantized representations for numbers and the straight through estimator (left) for the weight quantization and (right) the ReLU activation function.

4.2.1 Optimizing quantized RNNs for energy reconstruction

The simple RNN with 8 units and a sequence length of 5 is used to study quantized training. In previous work [2], PTQ was used to find the appropriate bitwidths for the weights of this network. A minimum of 14 total bits with three bits reserved for the integer part are found to be necessary. Lower bitwidths result in unacceptable loss in performance. Thus PTQ does not allow to take advantage of the four multiplications mode of the Agilex DSP.

Figure 4.3 shows the RMSE as function of the total bit width for different quantization methods for simple RNN in comparison to OFMax and the floating-point precision simple RNN. The RMSE for the quantized networks is evaluated using a simulation of a firmware implementation performed with the High Level Synthesis (HLS) language as explained in [1]. One can see the loss of performance for PTQ for a bit width lower than 14. A QAT is performed for the simple RNN network using QKeras [3]. The QAT reaches the full floating-point precision implemented in software starting at a bit width of 8. This makes QAT a suitable method to use the Agilex DSP in the four 9x9 multiplication mode and thus greatly reduces the resources needed in the FPGA. The implementation is able to use the 9x9 mode for the recurrent kernel multiplications which forms the majority of the multiplications in a RNN, however the input data is quantized to 16 bit precision which requires the kernel multiplication to use a higher bitwidth mode of the DSPs.

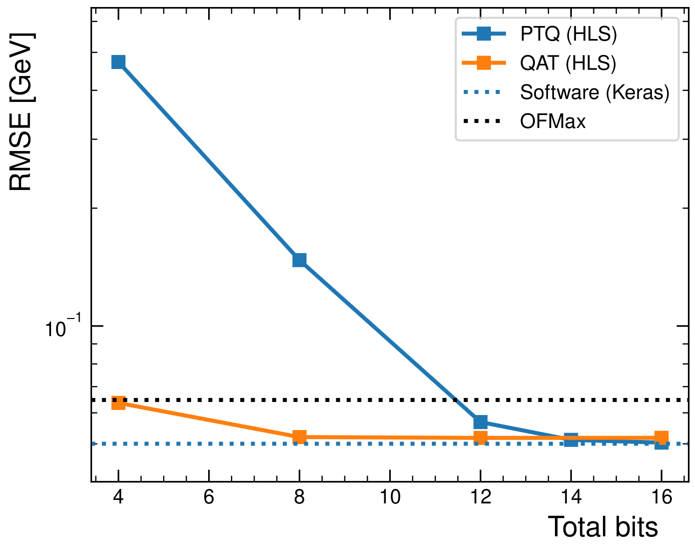


Figure 4.3: RMSE as function of the number of bits for a simple RNN with 8 units and a sequence length of 5 quantized with the PTQ and QAT methods, compared to the results from OFMax and the simple RNN with full floating point precision in software. The integer part is 3 bits for PTQ and 1 bit for QAT.

4.3 Pruned Neural Networks

4.3.1 Pruning

Pruning refers to the process of removing insignificant parameters from a neural network [66]. Pruning has been a popular model compression method for deployment on hardware since it can reduce the resource consumption due to the ability of avoiding performing multiplications for the removed parameters [67]. Weight pruning refers to a pruning method that targets only the weights of the neural network which results in sparse weight matrices. One way to achieve this is to implement a scheduled removal of low magnitude weights since weights that are very small only add small contributions to the model output and they can be removed in the pruning process. During the scheduled removal a small fraction of the weights are removed first, these weights are then kept at zero for the rest of the training. The training of the model then continues and in the next iteration more low magnitude weights are removed. Ultimately a predetermined final sparsity is reached and the pruning is stopped while the model can still be trained further to minimize potential performance drop caused by having fewer weights contributing to the output.

The effect of weight pruning can be seen in figure 4.4 which shows the distribution of weights for the 32 unit simple RNN with a sequence length of 5 obtained in section 3.3. The RNN is pruned to different sparsities using a polynomial pruning schedule. The original model has most of its weight in the lower bins, but not many in the zero bin. When pruned to 50% the model starts to have either the pruned weights at the zero bin or weights $|w| > 0.1$, this is also true for the 80% sparsity where most of the non-zero weights are $|w| > 0.3$. The model pruned to 95% sparsity follows similar trend with the absolute weight values being either zero or $|w| > 0.5$.

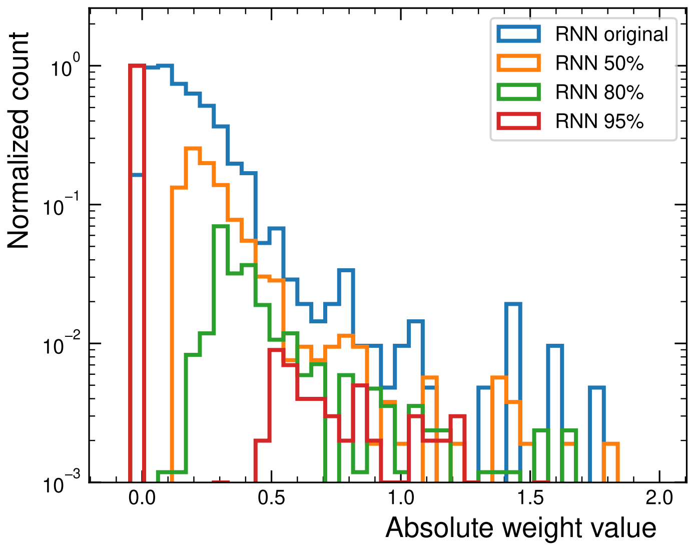


Figure 4.4: Distribution of weights for 32 unit RNN with different percentages of weights pruned.

Figure 4.5 shows the performance of the RNNs with 8, 16 and 32 units with a sequence length of 5, acquired in section 3.3, pruned to different sparsities compared to OFMax. All RNNs have significant performance drop when pruned. For the 8 unit RNN the performance drop is the most significant, at 50% sparsity the performance drops, but remains better than OFMax. However at 80% sparsity the performance is not as well as for OFMax and quickly degrades even more at 95% sparsity. The performance drop is lower for the RNNs with 16 and 32 units as expected. However, at the same number of non-zero weights, un-pruned small RNNs perform better than pruned large RNNs as can be seen in figure 4.5.

One can conclude that pruning does not have any advantage for the energy computation in the LAr calorimeter. It does not allow to improve the performance at a fixed number of required multiplications. Furthermore, if pruning is applied then multiplexing in the firmware, which allows the use of the same neural network instance in the hardware for several calorimeter channels [2], would not be possible anymore since the structure of the networks will depend on the pruning and thus an independent RNN instance is required per channel.

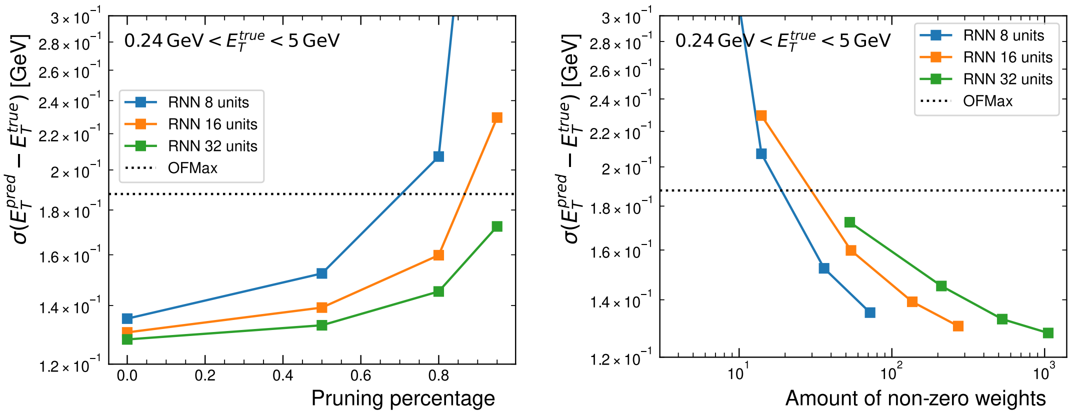


Figure 4.5: (left) Transverse energy resolution as function of the pruning sparsity for different simple RNNs with a sequence length of 5 compared to OFMax. (right) Transverse energy resolution as function of amount of non-zero weights for different simple RNNs with a sequence length of 5 compared to OFMax.

4.4 Resource efficient out-of-time pile-up correction

Section 3.3 shows that the effect of out-of-time pileup can be mitigated by having longer sequences with past information. However this comes at high computational cost, as each timestep increases the computational requirements. This section improves on the implementation by incorporating past information with minimal resource cost. Adding 25 timesteps requires adding 25 RNN cells for each energy computation while only contributing to small correction for out-of-time pileup. Figure 4.6 shows the architecture of the network that uses a fully connected layer with 16 units to process the first 25 samples of the sequence. This results in a vector of length 16 which is then used as the initial hidden state of a simple RNN layer with 16 units. This RNN layer then processes a sequence of length 5 which has one sample before the energy deposit and four around the peak. The model is then trained in full precision and QAT with all weights and biases quantized to 9 bits to evaluate if the Agilex DSP scheme can be utilized.

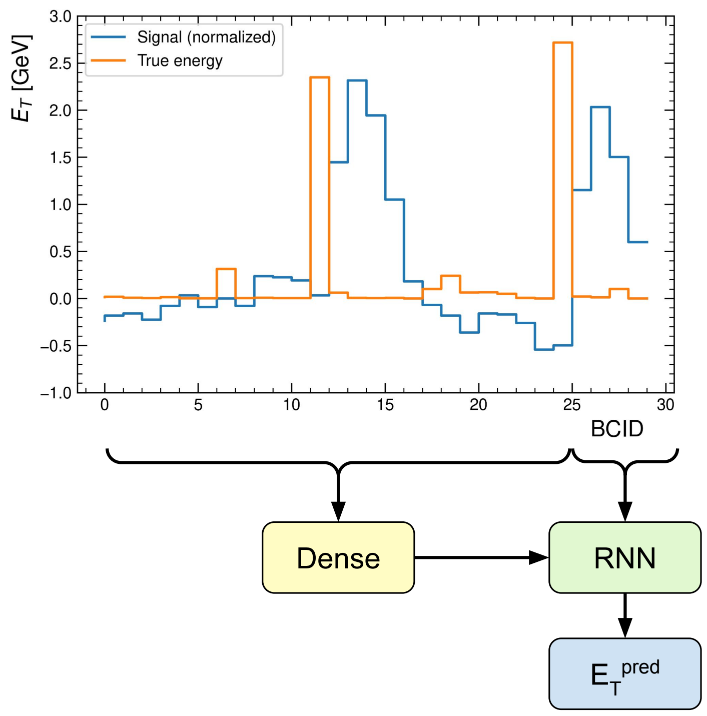


Figure 4.6: Correcting for out-of-time pileup by using a dense layer which sets the initial hidden state of the RNN.

Figure 4.7 shows the energy resolution as a function of the time gap to previous high energy deposit for this new architecture. The model is able to mitigate the drop of performance for overlapping events while keeping good performance for isolated pulses. The quantized model is performing as good as the full precision model. Figure 4.8 shows the energy resolution in the high and low gap region for the new architecture compared to simple RNNs and OFMax. In the low gap region the new architecture performs as good as the RNN with a sequence length of 30. In the high gap region the performances are similar for all models. Table 4.1 shows the required number of multiplications for each network. The architecture with a dense layer for out-of-time pileup correction significantly reduces the required amount of computations with respect to a simple RNN with sequence length of 30 which has the same amount of samples for out-of-time pileup correction.

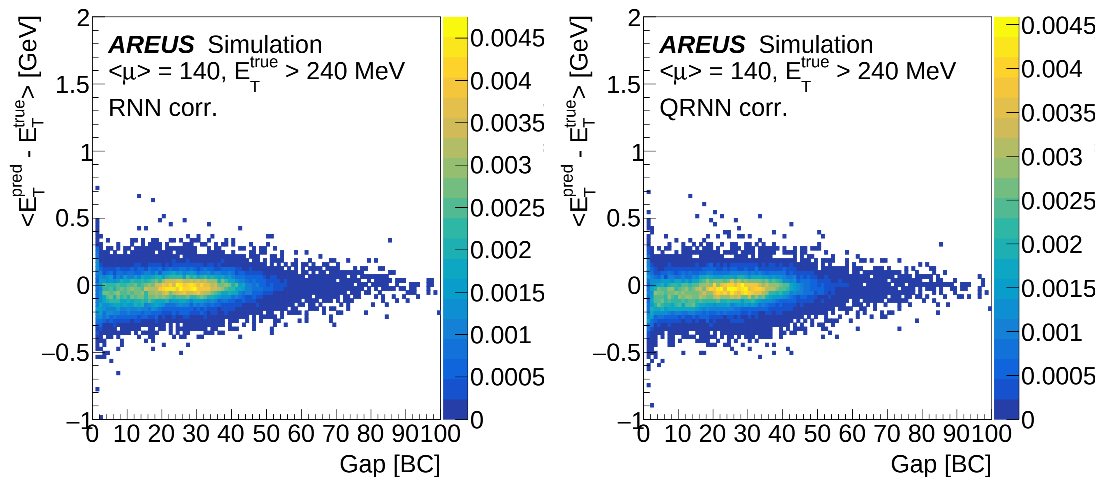


Figure 4.7: Transverse energy resolution as function of the time gap to previous high energy deposit for a simple RNN with 16 units and a sequence length of 5 that is initialized by a dense layer that probes the past information from 25 additional samples in the past. (left) The RNN trained with full floating point precision and (right) the network trained with QAT with a bit width of 9.

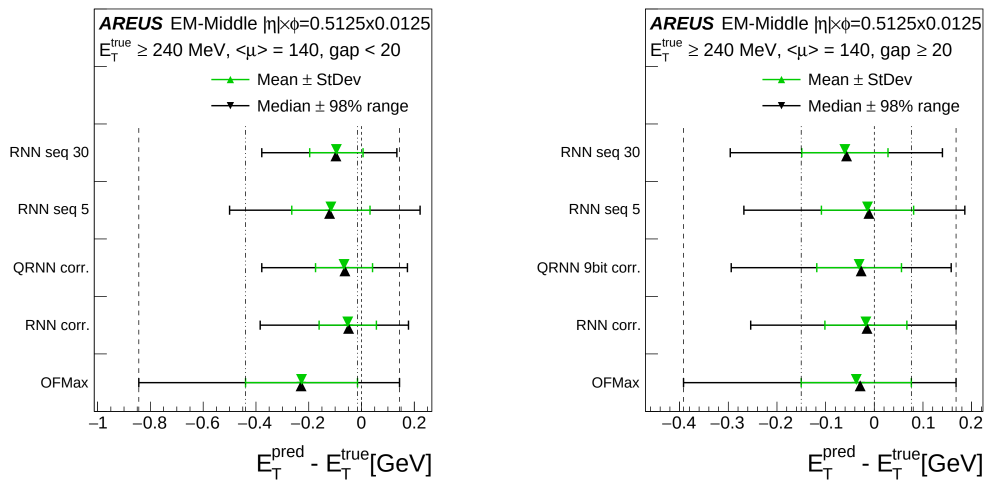


Figure 4.8: Transverse energy resolution for a simple RNN with 16 units and a sequence length of 5 that is initialized by a dense layer that probes the past information from 25 additional samples in the past (RNN corr.) and the same network quantized with a bit width of 9 (QRNN corr.) in comparison to a 16 unit simple RNN of sequence lengths 5 (RNN seq 5) and 30 (RNN seq 30) as well as OFMax for for (left) low gap and (right) high gap.

Model	RNN 16 units, seq 5	RNN 16 units, seq 30	RNN 16 units corr.
Number of multiplications	1376	8176	1776

Table 4.1: Number of multiplication needed for a simple RNN with 16 units with a sequence length of 5 or 30 and for the RNN with a sequence length of 5 that is initialized by a dense layer that probes the past information from 25 additional samples in the past (referred to as RNN 16 units corr.).

4.5 Conclusion

Deploying the RNNs on the FPGAs requires special considerations and optimizations for accurate real-time predictions. In order to reduce resources in the FPGAs, fixed-point arithmetic with the smallest possible bit width is required. Training the RNNs with floating point precision and quantizing the results after the training for FPGA implementation led to drop of accuracy due to quantization. This quantization post training also requires a relatively large number of bits to keep a reasonable accuracy. Training the network with quantized weight allows to mitigate this problem. It allows to recover the same accuracy with almost half the number of bits and thus greatly reduces the resources required in the FPGA.

Pruning neural networks allows to create sparse weight matrices that have mostly zeros. Multiplication with zero-weights can be omitted in the firmware implementation and thus reduce resource usage. However it has been seen that at an equal number of required multiplications smaller unpruned networks perform better than large pruned networks. Therefore pruning is deemed not suitable for the energy reconstruction in the LAr calorimeter.

The main improvement of neural networks over the optimal filtering algorithms resides in a better energy resolution in case of overlapping pulses caused by out-of-time pileup. However this requires long sequences that increase the resource consumption of RNNs. An alternative architecture using a dense layer to initialize the hidden state of a simple RNN is used to reduce the required resources while taking advantage of samples in the past to correct for out-of-time pileup. This new architecture has similar performance to RNNs with long sequences while reducing the needed resources by a factor of 5.

5 Energy reconstruction in the full detector

Summary

5.1 Introduction

The training of the RNNs developed in chapter 3 requires simulated data that takes into account the detector response for one of the calorimeter cells. The detector response determines the pulse shape that is seen by the readout electronics and that is used to compute the energy deposited in the calorimeter. However the pulse shape varies depending on the calorimeter region and the cell size. Particularly, pulse shapes are known to vary for different layers and for cells at different pseudorapidity (η).

To use neural networks for the reconstruction of the energy deposited in all cells of the detector, it is essential to have a well performing neural network assigned to each cell. Both legacy algorithms and RNNs are sensitive to changes in the pulse shape and thus a different set of weights is a priory needed for each cell. While it would be possible to train an independent network for all 182 468 cells, this would lead to an unsustainable amount of training time.

To keep high performance in all cells with a limited amount of trainings, it is possible to use unsupervised machine learning methods to group cells that have similar pulse shapes and that can use the same neural network with a single set of weights. Both calibration pulses, taken with injected signals in the detector, and physics pulses, measured in real proton-proton collisions, are available for the current LAr electronics. While the detector electronics will change during the phase II upgrade, one can use the current pulse shapes to demonstrate that grouping cells is achievable to reduce the required number of RNNs to train.

5.2 Clustering process

Unsupervised learning can be used to identify regions of the detector that have similar response. By clustering together pulses based on their similarity, unsupervised learning algorithms can identify groups of detector cells that have similar properties. One identified cluster will then use the same set of weights, significantly reducing the amount of required training time. In addition to identifying cells with similar response, also cells with anomalous response can be detected. Cells with anomalous response might require further investigations and ultimately help in identifying bad calorimeter cells.

5.2.1 Dimensionality reduction with t-SNE

t-SNE [68], short for t-Distributed Stochastic Neighbor Embedding, is a machine learning algorithm used for dimensionality reduction and data visualization. The algorithm takes high-dimensional data and maps it to a low-dimensional space while preserving the pairwise similarities between data points as much as possible. Points that are similar in the high-dimensional space will be close to each other in the low-dimensional space.

t-SNE works by first calculating the pairwise similarities between data points in the high-dimensional space using a Gaussian kernel. It then defines a probability distribution over pairs of points in the low-dimensional space, and minimizes the difference between this distribution and the pairwise similarity distribution in the high-dimensional space using a gradient descent algorithm.

5.2.2 DBSCAN for Cluster Classification

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [69], is a clustering algorithm used in data analysis to group together points that are close to each other by distance. Euclidean distance in two dimensions is used in this work. DBSCAN works by first determining core points; core points are defined as any point that have at least n points within a radius ϵ from it. The algorithm then recursively expands the cluster by adding points to the cluster if they are reachable from that cluster within the radius ϵ. The final cluster consists of points that can be reached via other points on a path where each step is smaller than the radius ϵ. Points that neither fulfill the conditions of the core point nor are reachable by other points in all clusters are classified as anomaly points. Anomaly points will be referred to with a cluster index -1 in what follows. The main advantages of DBSCAN are that it is able to automatically determine the required number of clusters, its ability to handle non-linearly separable clusters and its ability to identify anomalous points.

5.3 Clustering of calibration pulses

The LAr calibration pulses contain 768 samples over 800 ns with a spacing of 1.04 ns. The calibration pulses are measured in-situ for almost all calorimeter cells. The missing pulses are due to faulty cells that are marked as bad cells. In what follows, only LAr cells in the EMB and EMEC are considered; The HEC and FCAL pulses are excluded. The clustering is performed separately for each of the four layers in the EMB and EMEC. This is due to the fact that cells in different layers are known to have different size and thus different impedance and pulse shapes.

First the calibration pulses which are represented in 768 dimensions, which correspond to the number of samples, are reduced to two dimensions using t-SNE. The resulting map in two dimensions is shown in figure 5.1. The points in these two dimensions are drawn with different colors depending on the η of the cell. One can notice a correlation between the formed clusters and the color of the cluster. This is expected since the cell size and thus the pulse shape changes with the η of the cell.

Figure 5.2 shows the result of clustering by DBSCAN on the two dimensional mapping by t-SNE. The amount of found clusters is varying depending on the layer. EMB layers have between 2 and 11 clusters. However for EMEC there are up to 46 clusters in layer 2. Figure 5.3 shows the pulse shapes per layer where the different clusters are shown in different colors. One can see that pulses in the same clusters have similar shapes which qualitatively validates the clustering method. In all cases very different pulse shapes are in different clusters, but in some cases two groups of similar pulses have been assigned to different clusters.

The clustering results in the η - ϕ plane is shown in figure 5.4. The results clearly show that the pulse shapes are dependent on η and are symmetric in ϕ. The combination of t-SNE and DBSCAN is able to recognize not only the dependence of the pulse shape on η but also detect anomalous regions, such as cluster 7 of EMB layer 0, for which the pulse shape differs from other cells in the same η.

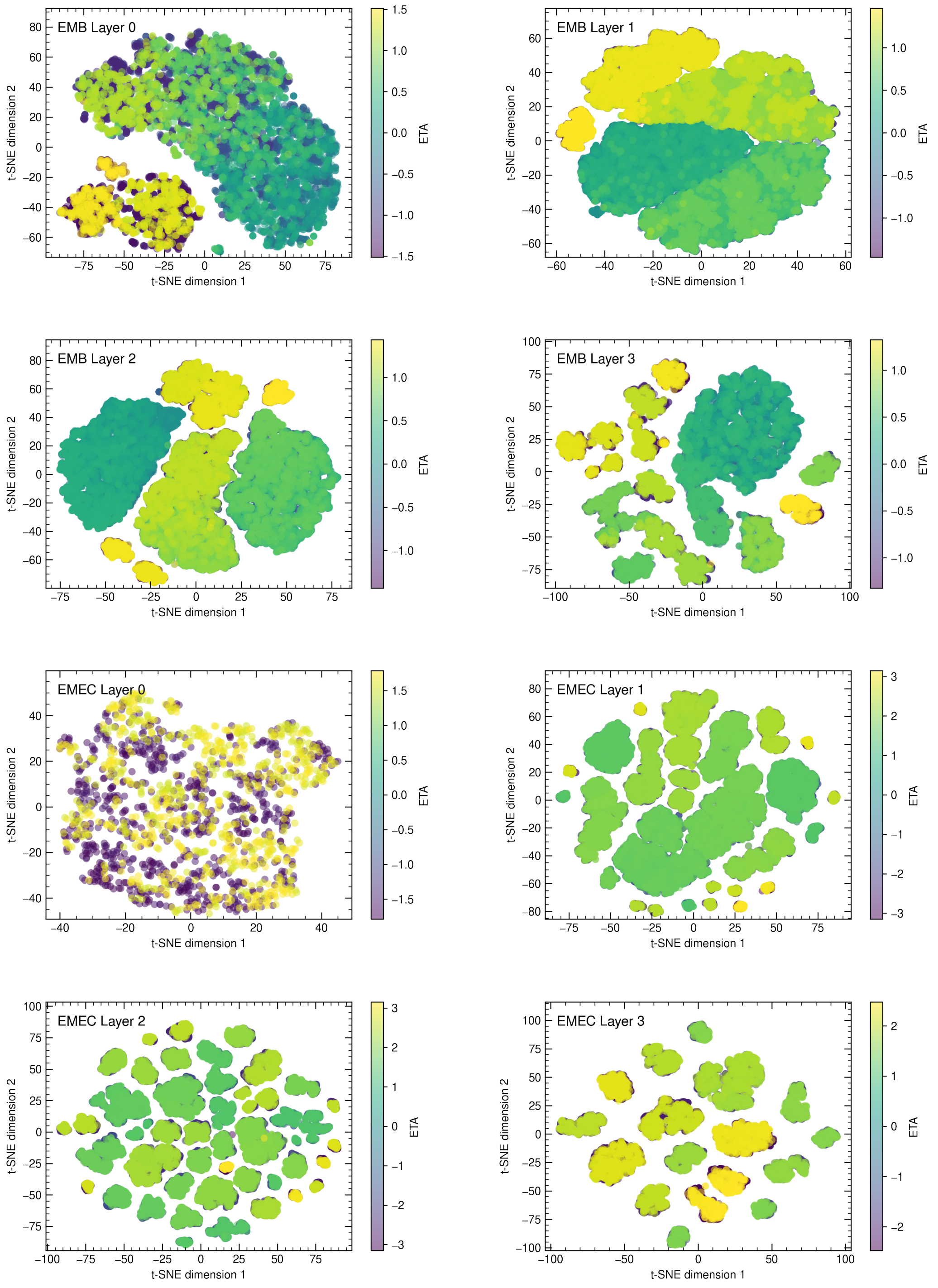


Figure 5.1: Distribution of the calorimeter cells as function of the two t-SNE dimensions for different layers in the EMB and EMEC. The color denotes the η of the cells.

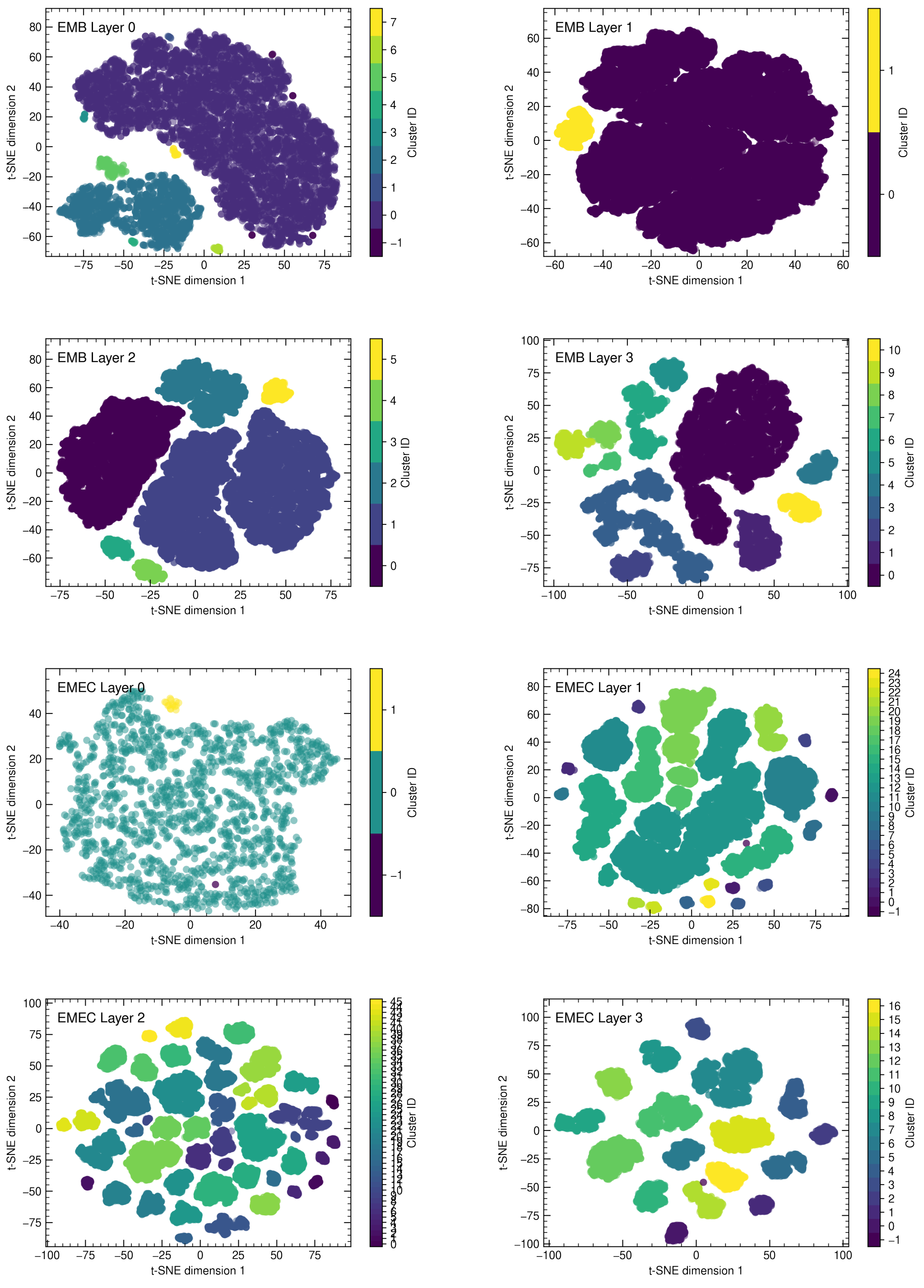


Figure 5.2: Distribution of the calorimeter cells as function of the two t-SNE dimensions for different layers in the EMB and EMEC. The color denotes the cluster ID as identified by DBSCAN.

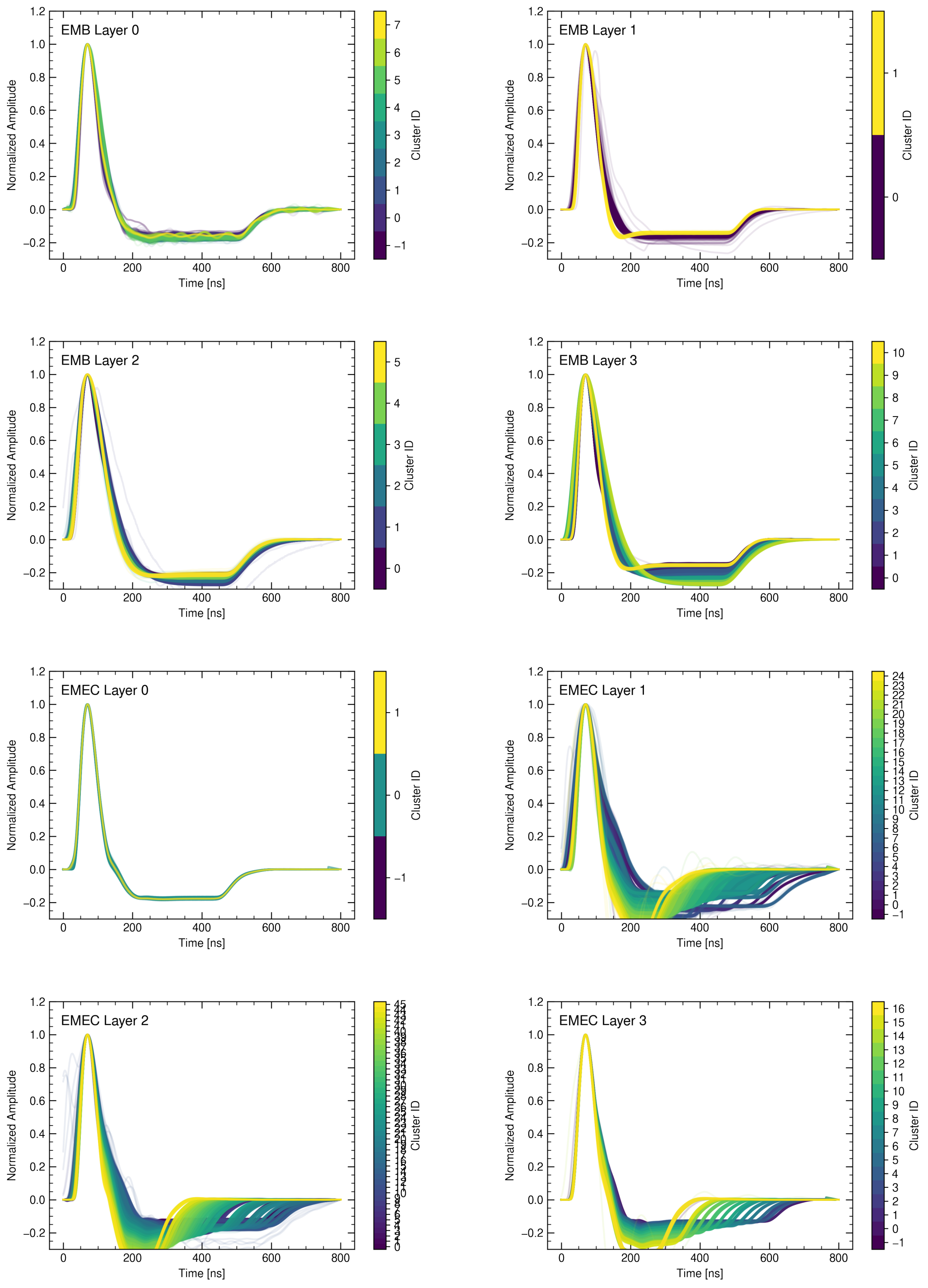


Figure 5.3: Calibration pulse shapes in the 4 layers of the EMB and EMEC. The colors represent the cluster ID as identified by DBSCAN.

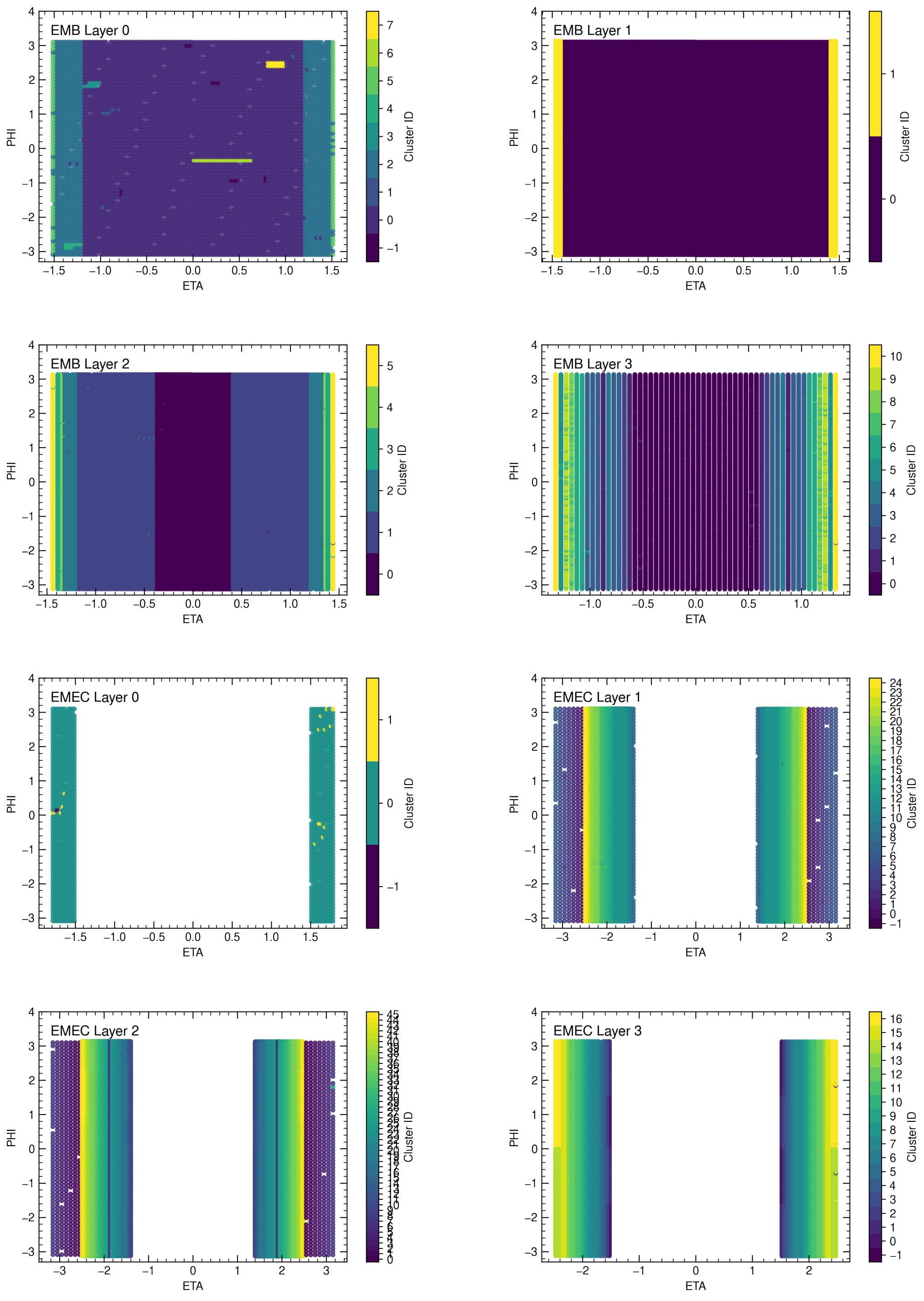


Figure 5.4: Distribution of the cluster ID in η - ϕ plane in the four layers of the EMB and EMEC.

5.4 RNN performance for clustered pulse shapes

To fully validate the clustering procedure, it is important to evaluate the performance of RNNs trained on clusters and compare them to RNNs trained on individual cells. The EMB layer 3 is chosen as a use case since it contains both clusters with similar pulse shapes and clusters with very different shapes. Figure 5.5 shows the performance of RNNs with different trainings for a cell in EMB layer 3 belonging to cluster 5. Three neural networks are compared: the first one is trained using data from cluster 5. In this case the pulse shape of the cell at the center of cluster 5 is used for the data simulation. This network is evaluated with data from another cell in the cluster and the same center cell it was trained on. The second network is trained on a different cluster randomly picked in EMB layer 3. The last network is trained on data that contains a mix of pulse shapes from all clusters in EMB layer 3.

One can see that the networks trained on the cluster center performs equally well with test data from another cell in the same cluster as it does with test data for the same cell. This validates that the clustering is able to group together cells with similar pulse shapes. The network trained on a different cluster shows very poor performance. Training on data containing a mixture of pulse shapes from all clusters does not recover the performance which validates the need for this clustering procedure.

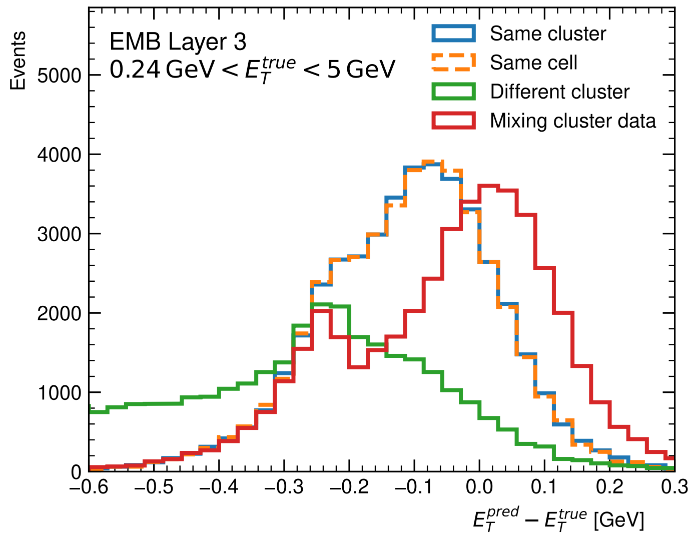
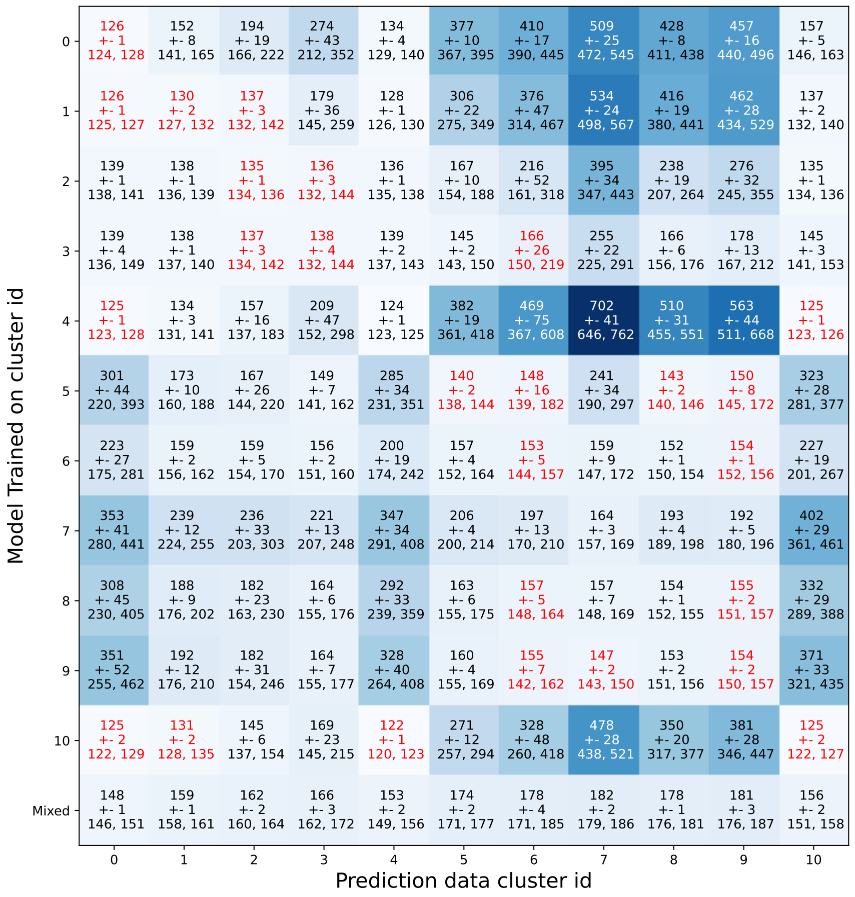


Figure 5.5: Transverse energy resolution for RNNs applied to a cell in EMB layer 3. In blue the RNN is tested on another cell from the same cluster. In orange the RNN is tested on the same cell that is used for training. In green the network is trained on a cell from a different cluster. In red the RNN is trained on a dataset that combines data from all clusters.

Figure 5.6 shows the performance from cross testing networks trained on data from each cluster in EMB layer 3 by picking the cell in the center of the cluster. Data used for evaluation was created for 10 randomly sampled cells of each cluster. The plot shows the mean and standard deviation of the and the standard deviation of $\sigma(E_T^{pred} - E_T^{true})$ for deposits with $E_T^{true} > 240$ MeV. In addition the minimum and maximum $\sigma(E_T^{pred} - E_T^{true})$ are shown. The best performing models are highlighted in red. The best models are chosen to be within one standard deviation of the training that have the best resolution.

For 8 out of 11 clusters the training performed on the same cluster is one of the best models. In the three other cases, models from different clusters perform better than the training on the same cluster. This could be due to fluctuations during the initialization of the RNN training or the cluster center not properly representing the pulse shapes in the cluster however the effect is not fully understood. The network trained with mixed data from all clusters gives mediocre performance and is in no case better than the one trained using the same cluster. This highlights the need to group cells for the training of the RNNs.



Model Trained on cluster id \ Prediction data cluster id	0	1	2	3	4	5	6	7	8	9	10
0	126 +- 1 124, 128	152 +- 8 141, 165	194 +- 19 166, 222	274 +- 43 212, 352	134 +- 4 129, 140	377 +- 10 367, 395	410 +- 17 390, 445	509 +- 25 472, 545	428 +- 8 411, 438	457 +- 16 440, 496	157 +- 5 146, 163
1	126 +- 1 125, 127	130 +- 2 127, 132	137 +- 3 132, 142	179 +- 36 145, 259	128 +- 1 126, 130	306 +- 22 275, 349	376 +- 47 314, 467	534 +- 24 498, 567	416 +- 19 380, 441	462 +- 28 434, 529	137 +- 2 132, 140
2	139 +- 1 138, 141	138 +- 1 136, 139	135 +- 1 134, 136	136 +- 3 132, 144	136 +- 1 135, 138	167 +- 10 154, 188	216 +- 52 161, 318	395 +- 34 347, 443	238 +- 19 207, 264	276 +- 32 245, 355	135 +- 1 134, 136
3	139 +- 4 136, 149	138 +- 1 137, 140	137 +- 3 134, 142	138 +- 4 132, 144	139 +- 2 137, 143	145 +- 2 143, 150	166 +- 26 150, 219	255 +- 22 225, 291	166 +- 6 156, 176	178 +- 13 167, 212	145 +- 3 141, 153
4	125 +- 1 123, 128	134 +- 3 131, 141	157 +- 16 137, 183	209 +- 47 152, 298	124 +- 1 123, 125	382 +- 19 361, 418	469 +- 75 367, 608	702 +- 41 646, 762	510 +- 31 455, 551	563 +- 44 511, 668	125 +- 1 123, 126
5	301 +- 44 220, 393	173 +- 10 160, 188	167 +- 26 144, 220	149 +- 7 141, 162	285 +- 34 231, 351	140 +- 2 138, 144	148 +- 16 139, 182	241 +- 34 190, 297	143 +- 2 140, 146	150 +- 8 145, 172	323 +- 28 281, 377
6	223 +- 27 175, 281	159 +- 2 156, 162	159 +- 5 154, 170	156 +- 2 151, 160	200 +- 19 174, 242	157 +- 4 152, 164	153 +- 5 144, 157	159 +- 9 147, 172	152 +- 1 150, 154	154 +- 1 152, 156	227 +- 19 201, 267
7	353 +- 41 280, 441	239 +- 12 224, 255	236 +- 33 203, 303	221 +- 13 207, 248	347 +- 34 291, 408	206 +- 4 200, 214	197 +- 13 170, 210	164 +- 3 157, 169	193 +- 4 189, 198	192 +- 5 180, 196	402 +- 29 361, 461
8	308 +- 45 230, 405	188 +- 9 176, 202	182 +- 23 163, 230	164 +- 6 155, 176	292 +- 33 239, 359	163 +- 6 155, 175	157 +- 5 148, 164	157 +- 7 148, 169	154 +- 1 152, 155	155 +- 2 151, 157	332 +- 29 289, 388
9	351 +- 52 255, 462	192 +- 12 176, 210	182 +- 31 154, 246	164 +- 7 155, 177	328 +- 40 264, 408	160 +- 4 155, 169	155 +- 7 142, 162	147 +- 2 143, 150	153 +- 2 151, 156	154 +- 2 150, 157	371 +- 33 321, 435
10	125 +- 2 122, 129	131 +- 2 128, 135	145 +- 6 137, 154	169 +- 23 145, 215	122 +- 1 120, 123	271 +- 12 257, 294	328 +- 48 260, 418	478 +- 28 438, 521	350 +- 20 317, 377	381 +- 28 346, 447	125 +- 2 122, 127
Mixed	148 +- 1 146, 151	159 +- 1 158, 161	162 +- 2 160, 164	166 +- 3 162, 172	153 +- 2 149, 156	174 +- 2 171, 177	178 +- 4 171, 185	182 +- 2 179, 186	178 +- 1 176, 181	181 +- 3 176, 187	156 +- 2 151, 158

Figure 5.6: $\sigma(E_T^{pred} - E_T^{true})$ for RNNs trained and applied to different clusters in EMB layer 3. Each model is trained with a cell in the center of the cluster and is evaluated on 10 randomly sampled cells in the cluster. For each box 4 values are given: the mean of $\sigma(E_T^{pred} - E_T^{true})$ over the 10 cells evaluated, its standard deviation, and the minimum and maximum $\sigma(E_T^{pred} - E_T^{true})$ for the 10 evaluated cells. The best performing models are highlighted in red. The best models are chosen to be within one standard deviation of the model that has the best resolution.

5.5 Clustering using physics pulses

Physics pulses, recorded during proton-proton collisions, differ from calibration pulses. Figure 5.7 shows an example of the calibration and physics pulse shape for the same calorimeter cell. The physics pulses should be ultimately used to train the neural networks. Therefore the clustering procedure that is demonstrated on calibration pulses is repeated for physics pulses. The measured physics pulses have 256 samples and are available for 90% of the cells. The missing pulses are due to problematic cells or problems with the data processing used to extract these pulses.

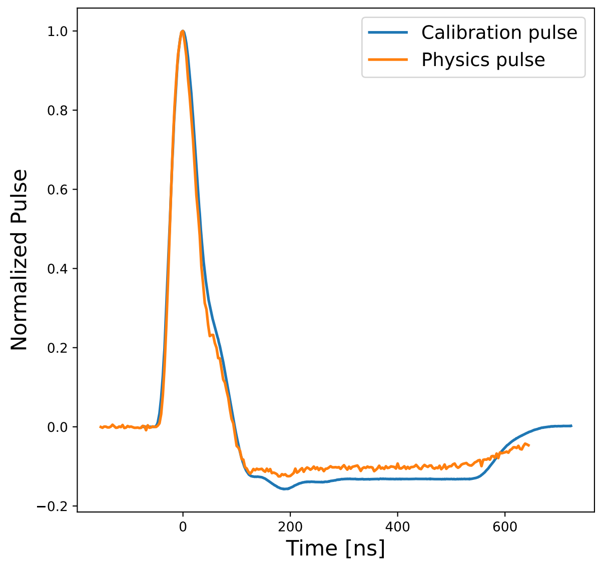


Figure 5.7: Calibration and physics pulse shapes for the same calorimeter cell.

5.5.1 Clustering process

The clustering process is the same as for the calibration pulses. Figure 5.8 shows a map of the calorimeter cells in the two dimensions produced by t-SNE. The colors represent the η of the cells. One can see a correlation between the η of the cells and the clusters that they belong to.

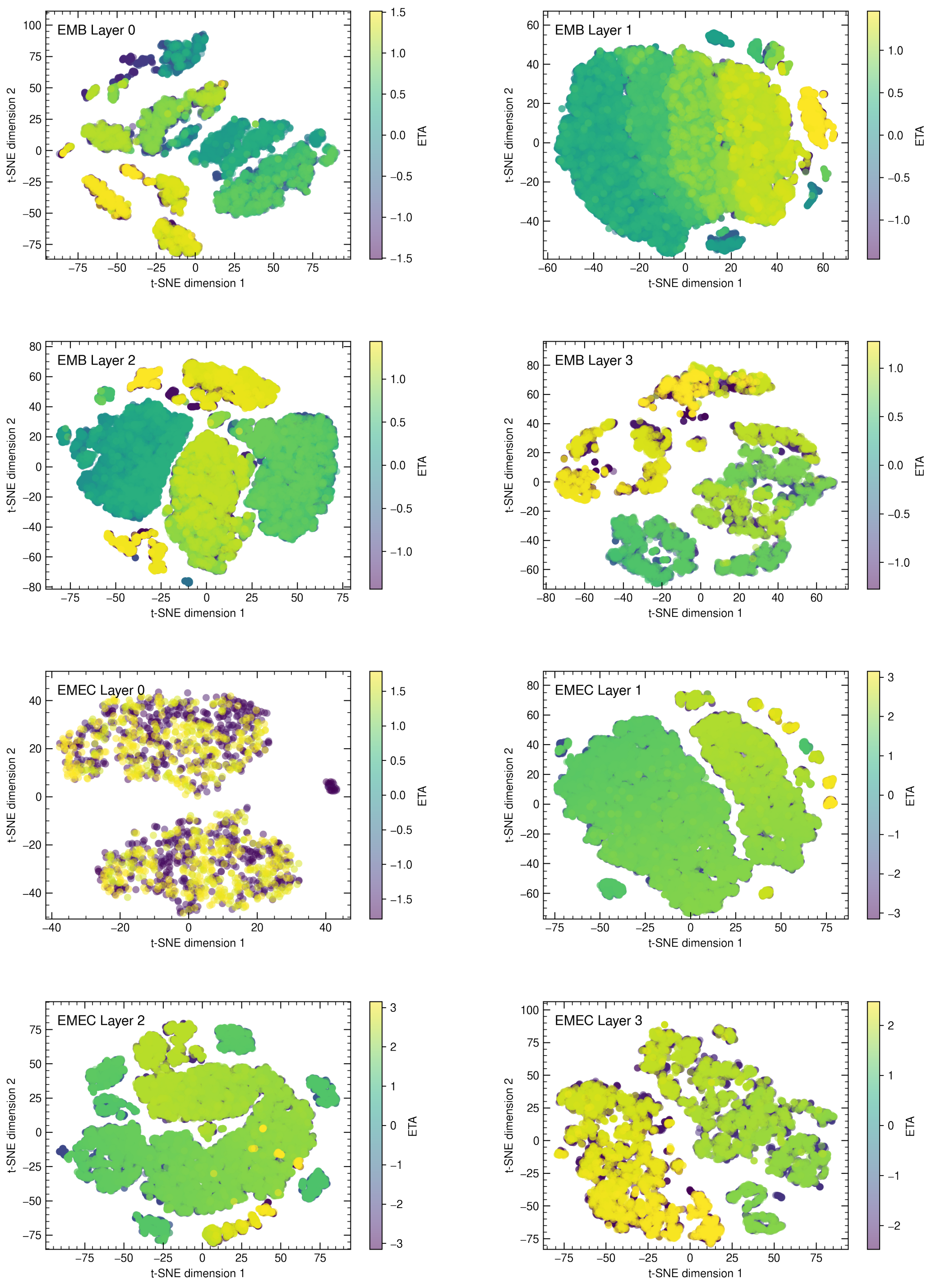


Figure 5.8: Distribution of the calorimeter cells as function of the two t-SNE dimensions for different layers in the EMB and EMEC for the physics pulses. The color denotes the η of the cells.

Figure 5.9 shows that DBSCAN yields approximately 10 clusters per layer, with the exception of EMEC layer 0. The number of clusters using the physics pulses is relatively smaller than the one using calibration pulses. Figure 5.10 shows the pulse shapes in the different clusters. There is less variation in the pulse shapes compared to calibration pulses.

Figure 5.11 shows a clear correlation between the η of the cell and the corresponding clusters. Clusters are found to span over larger η regions on average compared to calibration clusters. However physics pulses show more anomalous regions on the detector that break the correlation between the η and the pulse shape.

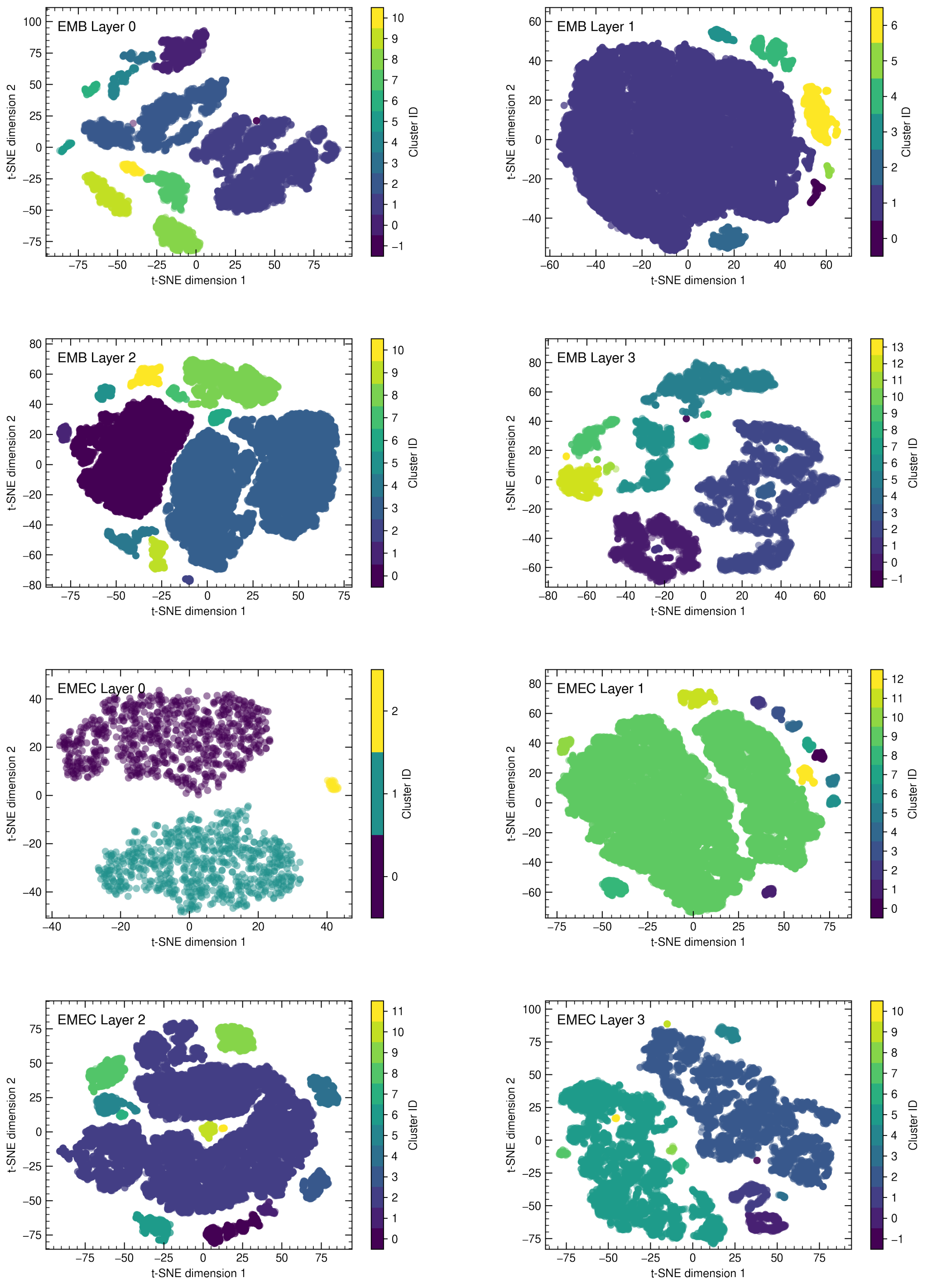


Figure 5.9: Distribution of the calorimeter cells as function of the two t-SNE dimensions for different layers in the EMB and EMEC for the physics pulses. The color denotes the cluster ID as identified by DBSCAN.

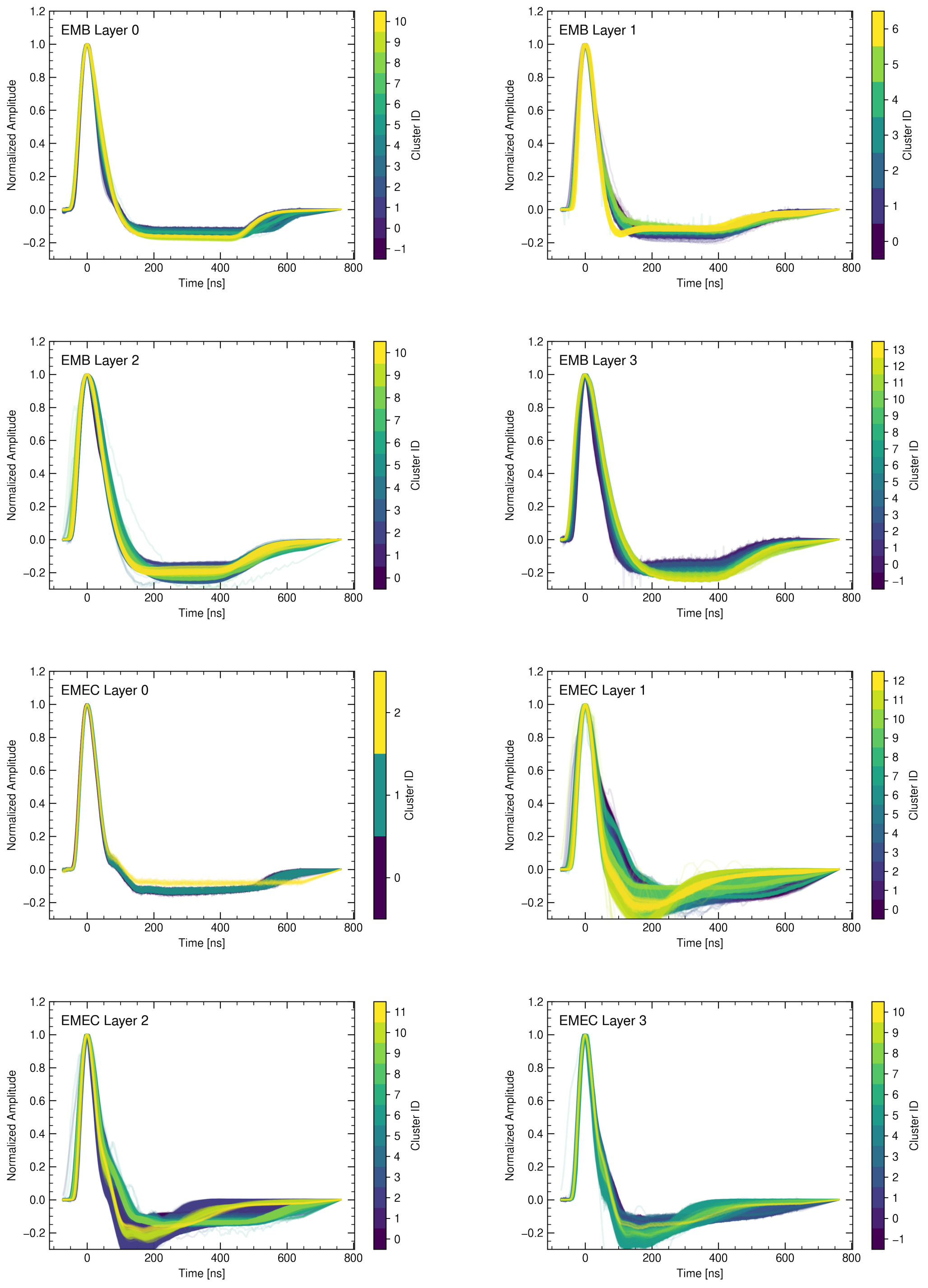


Figure 5.10: Physics pulse shapes in the 4 layers of the EMB and EMEC. The colors represent the cluster ID as identified by DBSCAN.

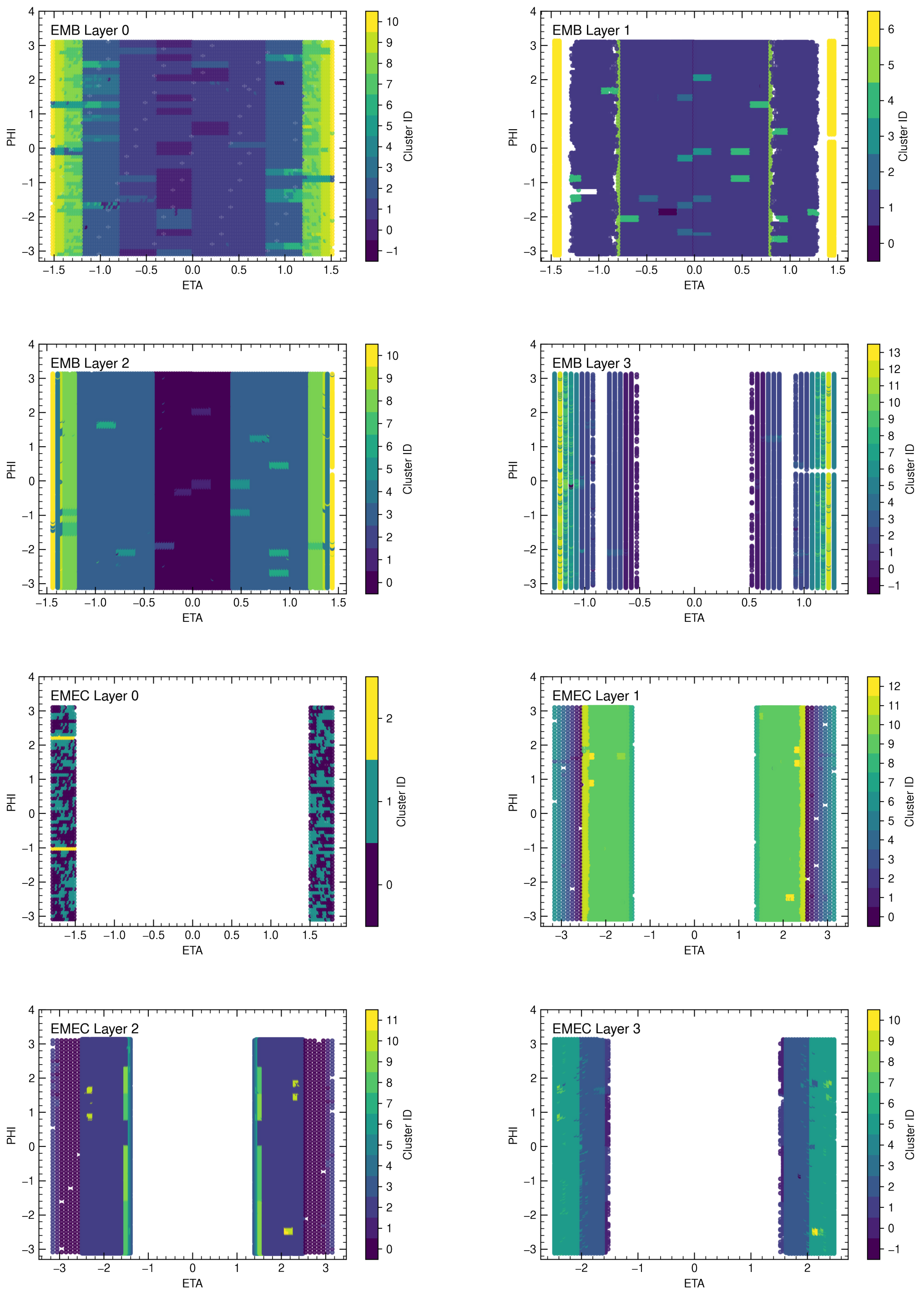


Figure 5.11: Distribution of the cluster ID in η - ϕ plane in the four layers of the EMB and EMEC for physics pulses.

5.6 Conclusion

The usage of RNNs for energy reconstruction in the full detector poses challenges to be able to extend the training of the neural networks to all 180000 calorimeter cells. Unsupervised learning and clustering algorithms can efficiently group LAr cells that have similar pulse shapes: t-SNE dimensionality reduction algorithm is used to create a two dimensional projection of the LAr cells based on the calibration pulses and DBSCAN clustering algorithm is then used to create clusters.The clustering algorithm is able to recover the dependence of the pulse shapes on η with remarkable accuracy.

The effectiveness of the clustering is validated by training RNNs on data that uses the pulse shapes from the cell at the center of the clusters. EMB layer 3 is chosen as a use case since it contains a variety of clusters with similar or very different pulse shapes. The RNNs perform well if they are trained on cells within the same cluster. RNNs trained on a mixture of cells from all clusters perform poorly thus validating the need for this clustering procedure. The clustering is only performed on EMB and EMEC cells; HEC and FCAL cells are left for future work. The clustering method is confirmed work for both calibration pulses and physics pulses. The clustering allows to reduce the number of needed neural networks to about 100 to cover all EMB and EMEC cells.

Conclusion

The unprecedented high luminosity of the HL-LHC leads to reduced resolution of the energy reconstructed in the ATLAS LAr calorimeter by the optimal filtering algorithm. To achieve the physics goals of the ATLAS experiment, new and improved energy reconstruction methods are required. The LAr calorimeter readout system will be replaced as part of the Phase-II upgrade in 2026-2028. The new electronics will be based on state-of-the-art FPGAs with increased computing capacity allowing to perform the energy reconstruction using neural networks.

This dissertation evaluates RNNs as candidates to replace the optimal filtering algorithm. Simple RNN and LSTM architectures are developed and are found to outperform the optimal filtering algorithm in HL-LHC conditions. Particularly, these RNNs are designed to accurately compute the energy of overlapping pulses, an area where the optimal filtering algorithm performs poorly. The RNNs are able to probe the corrections needed in case of pulse distortion due to overlapping pulses by incorporating information from past events prior to the energy deposit. Furthermore the RNNs are shown to be resilient against changes in the average pileup during the runs: RNNs trained on samples with an average pileup ranging between 100 and 200 have very good performance on any sample with average pileup in the same range. Finally, RNNs that can simultaneously compute the energy and the phase shift in the time of the pulse peak are developed. The phase shift is important to assign the deposited energy to the correct bunch crossing and to search for long lived particles. The RNNs are shown to outperform the optimal filtering algorithm both for the energy and phase shift reconstruction.

Deployment of the RNNs on FPGAs requires special considerations. In order to reduce resources in the FPGAs, fixed-point arithmetic with the smallest possible bit width is required. Training the network with quantized weights allows to reduce the number of bits by a factor of two while keeping similar accuracy to floating point precision. This leads to a drastic reduction of the resources needed in the FPGAs. Another way to reduce the computational resource is to prune the neural networks by removing insignificant weights. However it is shown that starting with small RNNs without pruning is more efficient than training large RNNs and then pruning them. Thus pruning is not considered further for the RNNs used for the energy computation in the LAr calorimeter. Finally a new architecture that uses a dense layer, that probes past information, to initialize the RNN is developed. This architecture is capable of reproducing the performance of RNNs with long sequences while requiring 5 times less computational power. All these developments make it possible to implement an efficient RNN on FPGAs.

To use the RNNs for the full LAr calorimeter, 182000 neural networks are needed.

Training such a large number of RNNs is practically impossible. To avoid this problem, unsupervised learning is used to automatically group calorimeter cells based on similarities in their electronic response. This method reduces the amount of required neural network to 100 for the EMB and EMEC part of the detector making the usage of RNNs more realistic for the full LAr calorimeter.

In summary RNNs are found to be excellent candidates to reconstruct the energy deposited in the LAr calorimeter during the HL-LHC era. They are found to outperform the current optimal filtering algorithm and are implementable on FPGAs. Additional work is still needed to further optimize the neural networks and to perform the final implementation in firmware. Furthermore, methods to calibrate the response of the RNNs in real data are still to be developed to take into account differences between the simulation data used in the training and real data at the HL-LHC.

Bibliography

[1] Georges Aad, Anne-Sophie Berthold, Thomas Calvet, et al. “Artificial Neural Networks on FPGAs for Real-Time Energy Reconstruction of the ATLAS LAr Calorimeters”. In: *Computing and Software for Big Science* 5.1 (Oct. 2021), p. 19. ISSN: 2510-2044. DOI: 10.1007/s41781-021-00066-y. URL: https://doi.org/10.1007/s41781-021-00066-y (cit. on pp. 23, 25, 72, 73, 97–100, 111, 114).

[2] G. Aad, T. Calvet, N. Chiedde, et al. “Firmware implementation of a recurrent neural network for the computation of the energy deposited in the liquid argon calorimeter of the ATLAS experiment”. In: *Journal of Instrumentation* 18.05 (May 2023), P05017. DOI: 10.1088/1748-0221/18/05/P05017. URL: https://dx.doi.org/10.1088/1748-0221/18/05/P05017 (cit. on pp. 23, 24, 111, 113, 116).

[3] Claudionor N. Coelho, Aki Kuusela, Shan Li, et al. “Automatic heterogeneous quantization of deep neural networks for low-latency inference on the edge for particle detectors”. In: *Nature Machine Intelligence* 3.8 (June 2021), pp. 675–686. DOI: 10.1038/s42256-021-00356-5. URL: https://doi.org/10.1038/s42256-021-00356-5 (cit. on pp. 25, 113, 114).

[4] Alessandro De Angelis and Mário Pimenta. *Introduction to Particle and Astroparticle Physics*. 2nd ed. Springer International Publishing, 2018. ISBN: 978-3-319-78181-5 (cit. on pp. 31, 32).

[5] *Standard Model of Elementary Particles*. https://commons.wikimedia.org/wiki/File:Standard_Model_of_Elementary_Particles.svg (cit. on p. 31).

[6] G. Aad, T. Abajyan, B. Abbott, et al. “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”. In: *Physics Letters B* 716.1 (Sept. 2012), pp. 1–29. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.020. URL: http://dx.doi.org/10.1016/j.physletb.2012.08.020 (cit. on p. 32).

[7] S. Chatrchyan, V. Khachatryan, A.M. Sirunyan, et al. “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC”. In: *Physics Letters B* 716.1 (Sept. 2012), pp. 30–61. ISSN: 0370-2693. DOI: 10.1016/j.physletb.2012.08.021. URL: http://dx.doi.org/10.1016/j.physletb.2012.08.021 (cit. on p. 32).

[8] Lyndon Evans and Philip Bryant. “LHC Machine”. In: *Journal of Instrumentation* 3.08 (Aug. 2008), S08001–S08001. DOI: 10.1088/1748-0221/3/08/s08001. URL: https://doi.org/10.1088%2F1748-0221%2F3%2F08%2Fs08001 (cit. on pp. 33, 43).

[9] Ewa Lopienska. “The CERN accelerator complex, layout in 2022. Complexe des accélérateurs du CERN en janvier 2022”. In: (2022). General Photo. URL: https://cds.cern.ch/record/2800984 (cit. on p. 34).

[10] Walter Scandale. “Proton–Proton and Proton–Antiproton Colliders”. In: *Reviews of Accelerator Science and Technology* 07 (2014), pp. 9–33. DOI: 10.1142/S1793626814300023 (cit. on p. 34).

[11] *Public atlas luminosity results for run-3 of the LHC.* URL: https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResultsRun3 (cit. on pp. 35, 36).

[12] Ivana Hristova on behalf of the ATLAS Collaboration. “Future Plans of the ATLAS Collaboration for the HL-LHC”. In: (2018) (cit. on p. 36).

[13] Georges Aad, N. de Groot, Frank Filthaut, et al. “The ATLAS Experiment at the CERN Large Hadron Collide”. In: *Journal of Instrumentation* 3 (2008) (cit. on pp. 37, 39, 40, 42, 44, 45).

[14] *Illustration of pseudorapidity.* https://tikz.net/axis2d_pseudorapidity/ (cit. on p. 39).

[15] S Haywood, L Rossi, R Nickerson, et al. *ATLAS inner detector: Technical Design Report, 2.* Technical design report. ATLAS. Geneva: CERN, 1997. URL: http://cds.cern.ch/record/331064 (cit. on p. 39).

[16] *Technical Design Report for the ATLAS Inner Tracker Pixel Detector.* Tech. rep. Geneva: CERN, 2017. DOI: 10.17181/CERN.FOZZ.ZP3Q. URL: http://cds.cern.ch/record/2285585 (cit. on p. 39).

[17] *Expected tracking and related performance with the updated ATLAS Inner Tracker layout at the High-Luminosity LHC.* Tech. rep. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-PHYS-PUB-2021-024. Geneva: CERN, 2021. URL: https://cds.cern.ch/record/2776651 (cit. on p. 40).

[18] *ATLAS liquid-argon calorimeter: Technical Design Report.* Technical design report. ATLAS. Geneva: CERN, 1996. DOI: 10.17181/CERN.FWRW.FOOQ. URL: https://cds.cern.ch/record/331061 (cit. on pp. 40, 42).

[19] Joao Pequenao. “Computer generated image of the ATLAS Liquid Argon”. 2008. URL: https://cds.cern.ch/record/1095928 (cit. on p. 41).

[20] D Banfi, M Delmastro, and M Fanti. “Cell response equalisation of the ATLAS electromagnetic calorimeter without the direct knowledge of the ionisation signals”. In: *Journal of Instrumentation* 1.08 (Aug. 2006), P08001. DOI: 10.1088/1748-0221/1/08/P08001. URL: https://dx.doi.org/10.1088/1748-0221/1/08/P08001 (cit. on p. 43).

[21] *Technical Design Report for the Phase-II Upgrade of the ATLAS TDAQ System.* Tech. rep. Geneva: CERN, 2017. DOI: 10.17181/CERN.2LBB.4IAL. URL: https://cds.cern.ch/record/2285584 (cit. on pp. 43, 45).

[22] G. Aad, A.V. Akimov, K. Al Khoury, et al. “The Phase-I trigger readout electronics upgrade of the ATLAS Liquid Argon calorimeters”. In: *Journal of Instrumentation* 17.05 (May 2022), P05024. DOI: 10.1088/1748-0221/17/05/P05024. URL: https://dx.doi.org/10.1088/1748-0221/17/05/P05024 (cit. on pp. 43, 100).

[23] *ATLAS tile calorimeter: Technical Design Report.* Technical design report. ATLAS. Geneva: CERN, 1996. DOI: 10.17181/CERN.JRBJ.7O28. URL: http://cds.cern.ch/record/331062 (cit. on p. 44).

[24] *Technical Design Report for the Phase-II Upgrade of the ATLAS Tile Calorimeter.* Tech. rep. Geneva: CERN, 2017. URL: https://cds.cern.ch/record/2285583 (cit. on p. 44).

[25] “ATLAS Liquid Argon Calorimeter Phase-II Upgrade Technical Design Report”. In: (2018) (cit. on pp. 45, 46).

[26] Signal Processing for the ATLAS Liquid ArgonCalorimeter : studies and implementation. “D. O. Damazio on behalf of the ATLAS liquid argon calorimeter group”. In: *2013 IEEE Nuclear Science Symposium and Medical Imaging Conference* (Nov. 2013) (cit. on pp. 47, 48).

[27] W.E.Cleland and E.G.Stern. “Signal processing considerations for liquid ionization calorimeters in a high rate environment”. In: *Nuclear Instruments and Methods in Physics Research* (July 1993) (cit. on p. 48).

[28] “ATLAS Liquid Argon Calorimeter Phase-I Upgrade Technical Design Report”. In: (2013) (cit. on p. 48).

[29] Steffen Stärz. “Energy Reconstruction and high-speed Data Transmission with FPGAs for the Upgrade of the ATLAS Liquid Argon Calorimeter at LHC”. PhD thesis. Dresden, Tech. U., 2015 (cit. on p. 48).

[30] Madysa, Nico. “AREUS: A Software Framework for ATLAS Readout Electronics Upgrade Simulation”. In: *EPJ Web Conf.* 214 (2019), p. 02006. DOI: 10.1051/epjconf/201921402006. URL: https://doi.org/10.1051/epjconf/201921402006 (cit. on p. 49).

[31] ATLAS Collaboration. *Athena.* Version 21.0.127. May 2021. DOI: 10.5281/zenodo.4772550. URL: https://doi.org/10.5281/zenodo.4772550 (cit. on p. 49).

[32] G. Aad, B. Abbott, D.C. Abbott, et al. "Electron and photon performance measurements with the ATLAS detector using the 2015-2017 LHC proton-proton collision data". In: *Journal of Instrumentation* 14.12 (Dec. 2019), P12006–P12006. DOI: 10.1088/1748-0221/14/12/p12006 (cit. on pp. 49–52).

[33] W Lampl, S Laplace, D Lelas, et al. *Calorimeter Clustering Algorithms: Description and Performance.* Tech. rep. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROUPS/PHYSICS/PUBNOTES/ATL-LARG-PUB-2008-002. Geneva: CERN, 2008. URL: https://cds.cern.ch/record/1099735 (cit. on p. 49).

[34] Morad Aaboud et al. "Electron reconstruction and identification in the ATLAS experiment using the 2015 and 2016 LHC proton-proton collision data at $\sqrt{s} = 13 TeV$ ". In: *Eur. Phys. J. C* 79.8 (2019), p. 639. DOI: 10.1140/epjc/s10052-019-7140-6. arXiv: 1902.04655 [physics.ins-det] (cit. on p. 51).

[35] T Cornelissen, M Elsing, I Gavrilenko, et al. "The new ATLAS track reconstruction (NEWT)". In: *Journal of Physics: Conference Series* 119.3 (July 2008), p. 032014. DOI: 10.1088/1742-6596/119/3/032014. URL: https://dx.doi.org/10.1088/1742-6596/119/3/032014 (cit. on p. 51).

[36] T G Cornelissen, M Elsing, I Gavrilenko, et al. "The global χ^2 track fitter in ATLAS". In: *Journal of Physics: Conference Series* 119.3 (July 2008), p. 032013. DOI: 10.1088/1742-6596/119/3/032013. URL: https://dx.doi.org/10.1088/1742-6596/119/3/032013 (cit. on p. 51).

[37] R Frühwirth. "A Gaussian-mixture approximation of the Bethe–Heitler model of electron energy loss by bremsstrahlung". In: *Computer Physics Communications* 154.2 (2003), pp. 131–142. ISSN: 0010-4655. DOI: https://doi.org/10.1016/S0010-4655(03)00292-3. URL: https://www.sciencedirect.com/science/article/pii/S0010465503002923 (cit. on p. 51).

[38] R. Frühwirth. "Application of Kalman filtering to track and vertex fitting". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 262.2 (1987), pp. 444–450. ISSN: 0168-9002. DOI: https://doi.org/10.1016/0168-9002(87)90887-4. URL: https://www.sciencedirect.com/science/article/pii/0168900287908874 (cit. on p. 51).

[39] Francois Chollet. *Deep Learning with Python.* Manning Publications, 2017. ISBN: 9781617294433 (cit. on pp. 55, 61).

[40] Sandro Skansi. *Introduction to Deep Learning.* Springer International Publishing, 2018. ISBN: 978-3-319-73004-2 (cit. on p. 56).

[41] Giuseppe Carleo, Ignacio Cirac, Kyle Cranmer, et al. "Machine learning and the physical sciences". In: (2019). cite arxiv:1903.10563. URL: http://arxiv.org/abs/1903.10563 (cit. on p. 56).

[42] F. Rosenblatt. *The perceptron - A perceiving and recognizing automaton.* Tech. rep. 85-460-1. Ithaca, New York: Cornell Aeronautical Laboratory, Jan. 1957 (cit. on p. 56).

[43] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. "Deep Sparse Rectifier Neural Networks". In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics.* Ed. by Geoffrey Gordon, David Dunson, and Miroslav Dudík. Vol. 15. Proceedings of Machine Learning Research. Fort Lauderdale, FL, USA: PMLR, Aug. 2011, pp. 315–323. URL: http://proceedings.mlr.press/v15/glorot11a.html (cit. on p. 58).

[44] A.L. Maas, A.Y. Hannun, and A.Y. Ng. "Rectifier Nonlinearities Improve Neural Network Acoustic Models". In: *Proceedings of the International Conference on Machine Learning.* 2013 (cit. on p. 59).

[45] Dan Hendrycks and Kevin Gimpel. *Gaussian Error Linear Units (GELUs).* 2023. arXiv: 1606.08415 [cs.LG] (cit. on p. 59).

[46] Sebastian Ruder. *An overview of gradient descent optimization algorithms.* 2017. arXiv: 1609.04747 [cs.LG] (cit. on pp. 63, 65).

[47] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *CoRR* abs/1412.6980 (2015) (cit. on p. 65).

[48] John Duchi, Elad Hazan, and Yoram Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html (cit. on p. 65).

[49] Geoffrey E. Hinton. *RMSprop.* URL: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf (cit. on p. 65).

[50] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. "Learning representations by back-propagating errors". In: *Nature* 323.6088 (Oct. 1986), pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0. URL: https://doi.org/10.1038/323533a0 (cit. on p. 65).

[51] Seppo Linnainmaa. "Taylor expansion of the accumulated rounding error". In: *BIT Numerical Mathematics* 16.2 (June 1976), pp. 146–160. ISSN: 1572-9125. DOI: 10.1007/BF01931367. URL: https://doi.org/10.1007/BF01931367 (cit. on p. 65).

[52] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* http://www.deeplearningbook.org. MIT Press, 2016 (cit. on pp. 65, 66).

[53] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. "Learning Internal Representations by Error Propagation". In: *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* Cambridge, MA, USA: MIT Press, 1986, pp. 318–362. ISBN: 026268053X (cit. on p. 66).

[54] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. *Unsupervised Learning of Video Representations using LSTMs.* 2016. arXiv: 1502.04681 [cs.LG] (cit. on p. 66).

[55] Jeffrey L. Elman. “Finding structure in time”. In: *Cognitive Science* 14.2 (1990), pp. 179–211. ISSN: 0364-0213. DOI: https://doi.org/10.1016/0364-0213(90)90002-E. URL: https://www.sciencedirect.com/science/article/pii/036402139090002E (cit. on p. 67).

[56] Paul J. Werbos. *Generalization of backpropagation with application to a recurrent gas market model.* Jan. 1988. DOI: 10.1016/0893-6080(88)90007-x. URL: https://doi.org/10.1016/0893-6080(88)90007-x (cit. on p. 68).

[57] Sepp Hochreiter and Jürgen Schmidhuber. “Long Short-Term Memory”. In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735 (cit. on p. 68).

[58] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks.* Vol. 385. Studies in Computational Intelligence. Springer, 2012, pp. 1–131. ISBN: 978-3-642-24796-5. URL: http://dx.doi.org/10.1007/978-3-642-24797-2 (cit. on p. 69).

[59] F.A. Gers, J. Schmidhuber, and F. Cummins. “Learning to forget: continual prediction with LSTM”. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470).* Vol. 2. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218 (cit. on p. 69).

[60] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics.* Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256. URL: https://proceedings.mlr.press/v9/glorot10a.html (cit. on p. 73).

[61] *JRJC 2021. Book of Proceedings.* Sept. 2022. URL: https://hal.science/hal-03832762 (cit. on p. 106).

[62] Siemens. *Algorithmic C (AC) Datatypes Reference Manual.* https://raw.githubusercontent.com/hlslibs/ac_types/master/pdfdocs/ac_datatypes_ref.pdf. 2022 (cit. on p. 111).

[63] *Intel® Agilex™ Variable Precision DSP Blocks User Guide.* https://www.intel.com/content/www/us/en/docs/programmable/683037/21-2/variable-precision-dsp-blocks-architecture.html. Accessed: 2023-03-20 (cit. on pp. 111, 112).

[64] Bert Moons, Koen Goetschalckx, Nick Van Berckelaer, et al. *Minimum Energy Quantized Neural Networks.* 2017. arXiv: 1711.00215 [cs.NE] (cit. on p. 112).

[65] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. *Estimating or Propagating Gradients Through Stochastic Neurons for Conditional Computation*. 2013. arXiv: 1308.3432 [cs.LG] (cit. on p. 113).

[66] Yann LeCun, John Denker, and Sara Solla. "Optimal Brain Damage". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 2. Morgan-Kaufmann, 1989. URL: https://proceedings.neurips.cc/paper_files/paper/1989/file/6c9882bbac1c7093bd25041881277658-Paper.pdf (cit. on p. 114).

[67] J. Duarte, S. Han, P. Harris, et al. "Fast inference of deep neural networks in FPGAs for particle physics". In: *Journal of Instrumentation* 13.07 (July 2018), P07027–P07027. DOI: 10.1088/1748-0221/13/07/p07027. URL: https://doi.org/10.1088/1748-0221/13/07/p07027 (cit. on p. 114).

[68] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html (cit. on p. 121).

[69] Martin Ester, Hans-Peter Kriegel, Jörg Sander, et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231 (cit. on p. 121).