DE LA RECHERCHE À L'INDUSTRIE

RAINFROG

AUTOMATIC DETECTOR RECOGNITION ON THE ATLAS/NSW FABRICATION FACILITY AT SACLAY

ROBUST AI NETWORK
FOR RECOGNITION OF OBJECTS ON GANTRY

RÉSEAU ARTISANAL D'INFÉRENCE NEURONALE
FACILITANT LA RECONNAISSANCE D'OBJETS, SANS GARANTIE

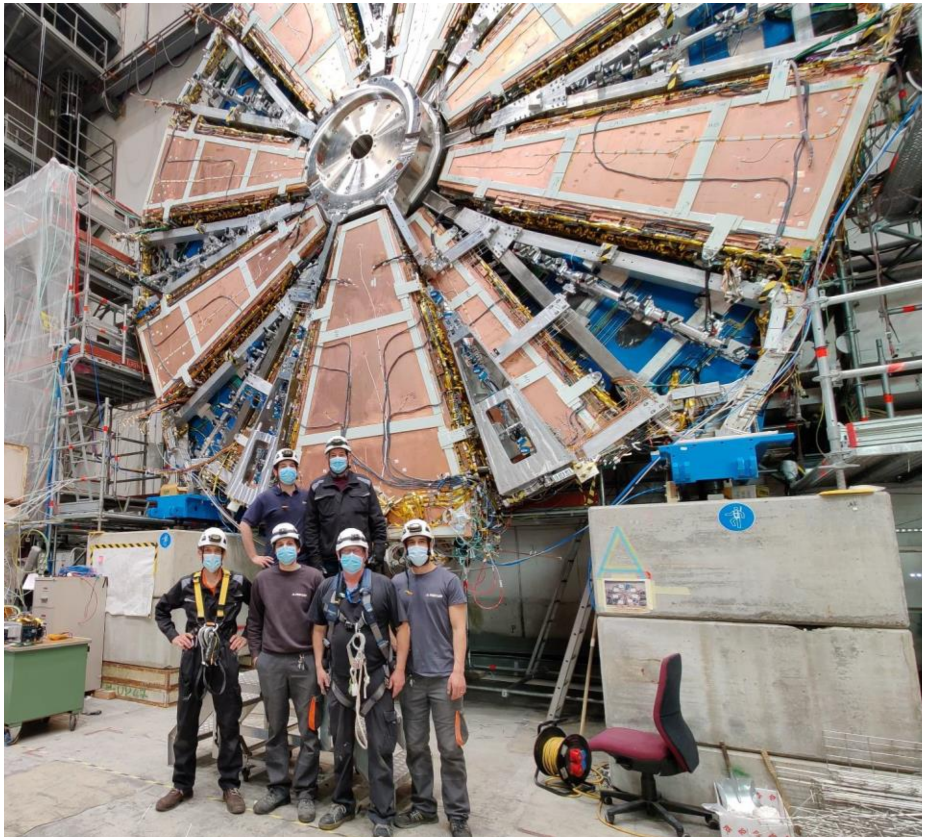
INSTITUT DE RECHERCHE SUR
LES LOIS FONDAMENTALES DE L'UNIVERS

Irfu

www.cea.fr

LHC upgrade LS2	2 detector « wheels », 10 m diameter
	32 Micromegas tracking detector modules
	4 different shapes
	4 different production sites
Status as of march '21	All 4 sites have terminated production
	1st wheel being assembled at CERN

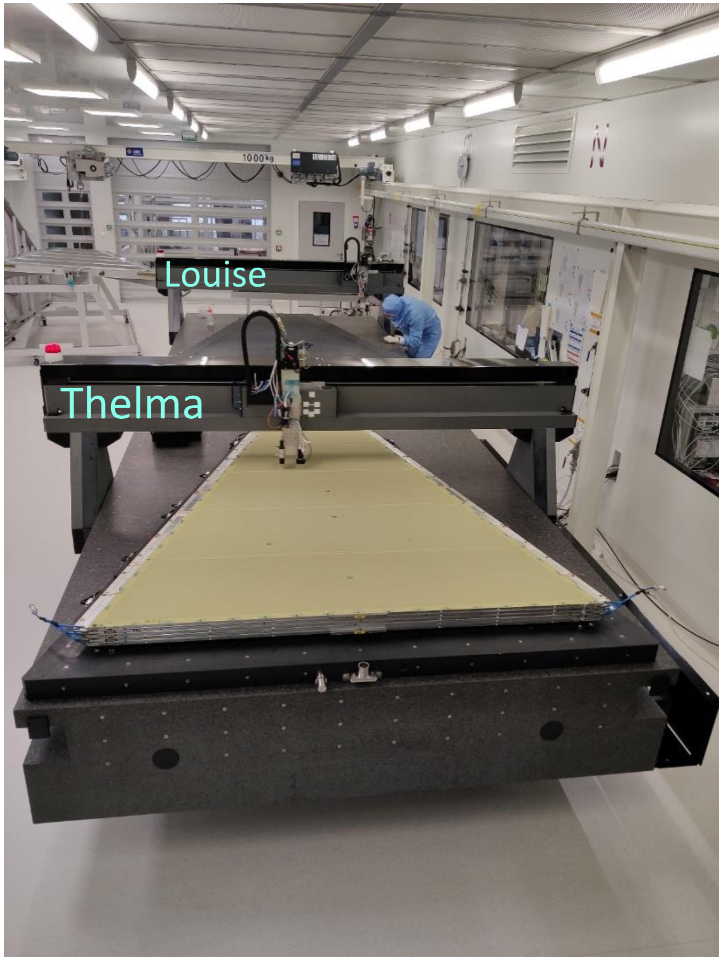
The modules built at Saclay (shape: Large Module 1)



Large and small « petals »

Improve process automation in the Saclay NSW facility

Saclay Irfu site context	Fabrication & test of detector elements in the CICLAD clean room facility at Saclay
	NSW LM1 Micromegas detectors are scanned for planarity on 2 instrumented gantry granite tables
	Planarity scans occur at various stages of construction and for final QC acceptance
	A large variety of scans requiring different settings
Problem	Can we automatically find the active configuration, just by looking at the current part with a camera?
	Algorithmic image processing would be very complicated. **Can we handle this by machine learning?**

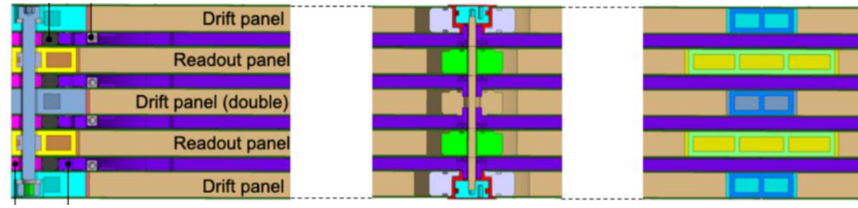


A completed module being scanned for planarity with a contactless optical sensor

A VARIETY OF PLANARITY SCANS (1/2, PANELS)

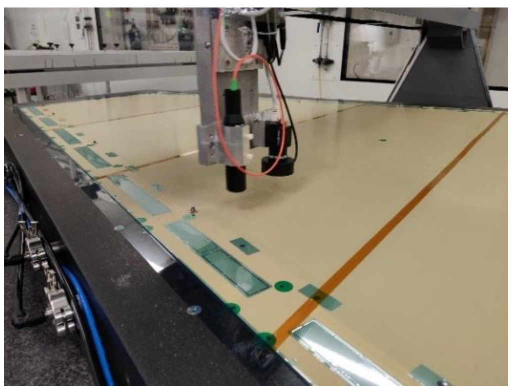
A Module is a sandwich of 5 composite PCB-Al Honeycomb-PCB Panels

DriftB
RdoutE
DriftC
RdoutS
DriftF

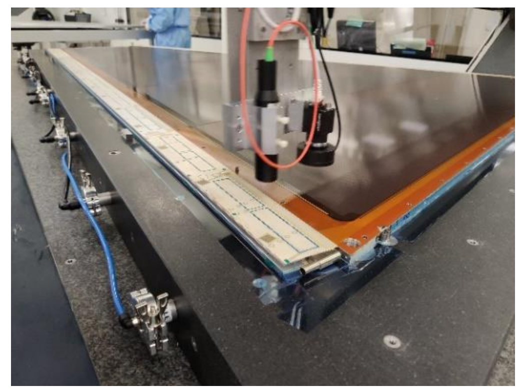


Each panel is scanned 6 times

- Prod step — AiPcb ①
- Prod step — AiHoney ②
- Prod step — BiPcb ③
- Final QC — Apumped ④
- Final QC — BonPads ⑤
- Final QC — AonPads ⑥

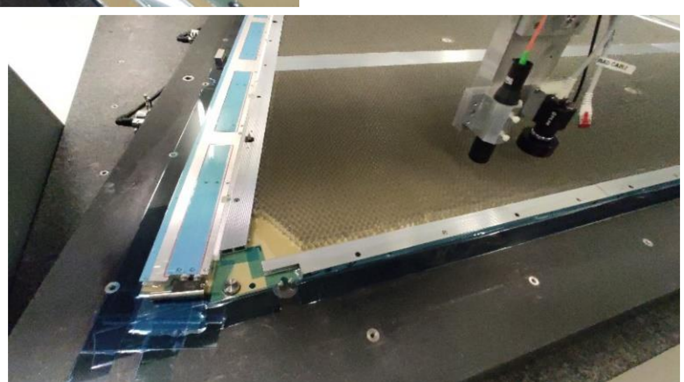


❶ ❸



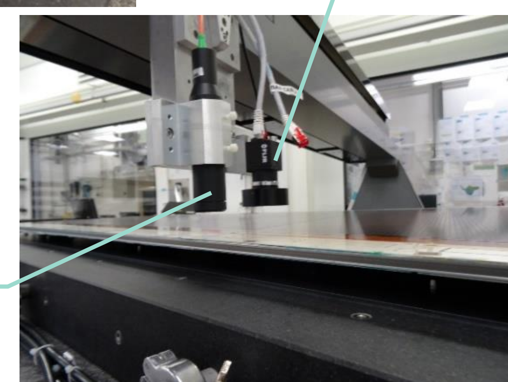
❹

❷



❺ ❻

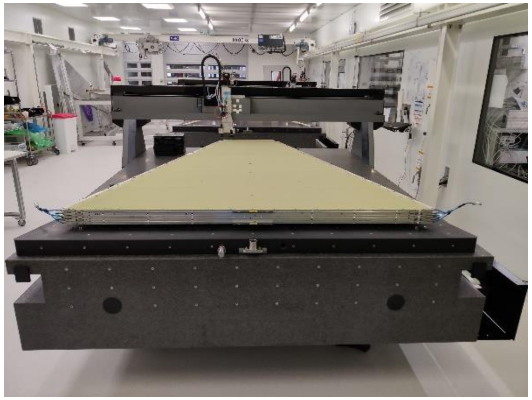
Camera



Confocal chromatic laser distance sensor

Module

Final QC ModulE



31 distinct scan configurations

5 panel types	*6 scans each*
1 module	*1 single scan*

Can we identify the current configuration automatically, just by looking at the object laid on the table?*

Machine Learning? Where to take photos? How to combine them?

** With a camera*

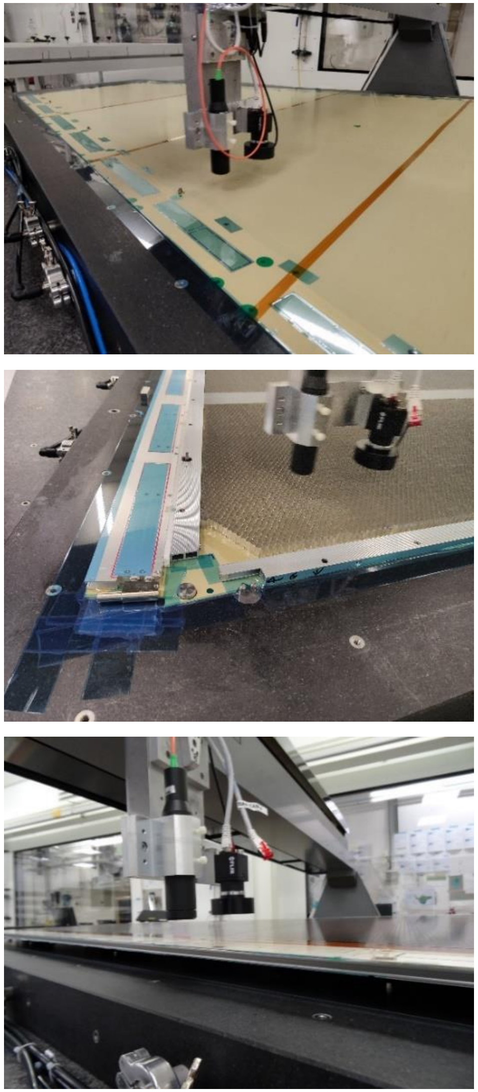
Look and guess

What needs to be discovered

- The **type** and **subtype** of the object: drift panel *front/central/back*, readout panel *eta/stereo*, module
- The nature of the **surface** *pcb/honeycomb/resist/copper*
- Which **side** of the part is exposed for measurement *A/B*
- The **height** at which the part is exposed

A non standard image classification problem

- A total of 20 categories in 5 **families**
- Will pick the highest score in each **family**
- ⚠ Categories are non exclusive, showing inclusion and intersection across **families**



A variety of configurations of parts, sides, surfaces, heights

7 OVERIMPOSED CLASSIFIERS IN THE SAME NETWORK

Instead of directly assigning the final configuration, make a prediction for intermediate family classes

20 different categories in 5 families	topc (3)	DriftP, RdoutP, Module
	type (6)	RdoutS, RdoutE, DriftF, DriftC, DriftB, modulE
	surf (5)	PcbInt, HoneyC, Resist, PcbExt, Cathod
	side (2)	Axposd, Bxposd
	zpos (4)	PcbSol, PnlSol, PnlPad, MdlPad

After fusion, if the 5 family predictions are found consistent, the final configuration will be assigned with a simple look up table

+9 control categories in 2 families	ploc (6)	photoA, photoB, photoC, photoD, photoE, photoF
	pcam (3)	MikeMi, SonyDS, FliCam

PATCH FUSION

❶

Take photos at 6 strategic locations

A	Pcb frontier & edge
B, C, E, F	Corners
D	Interconnect

❷

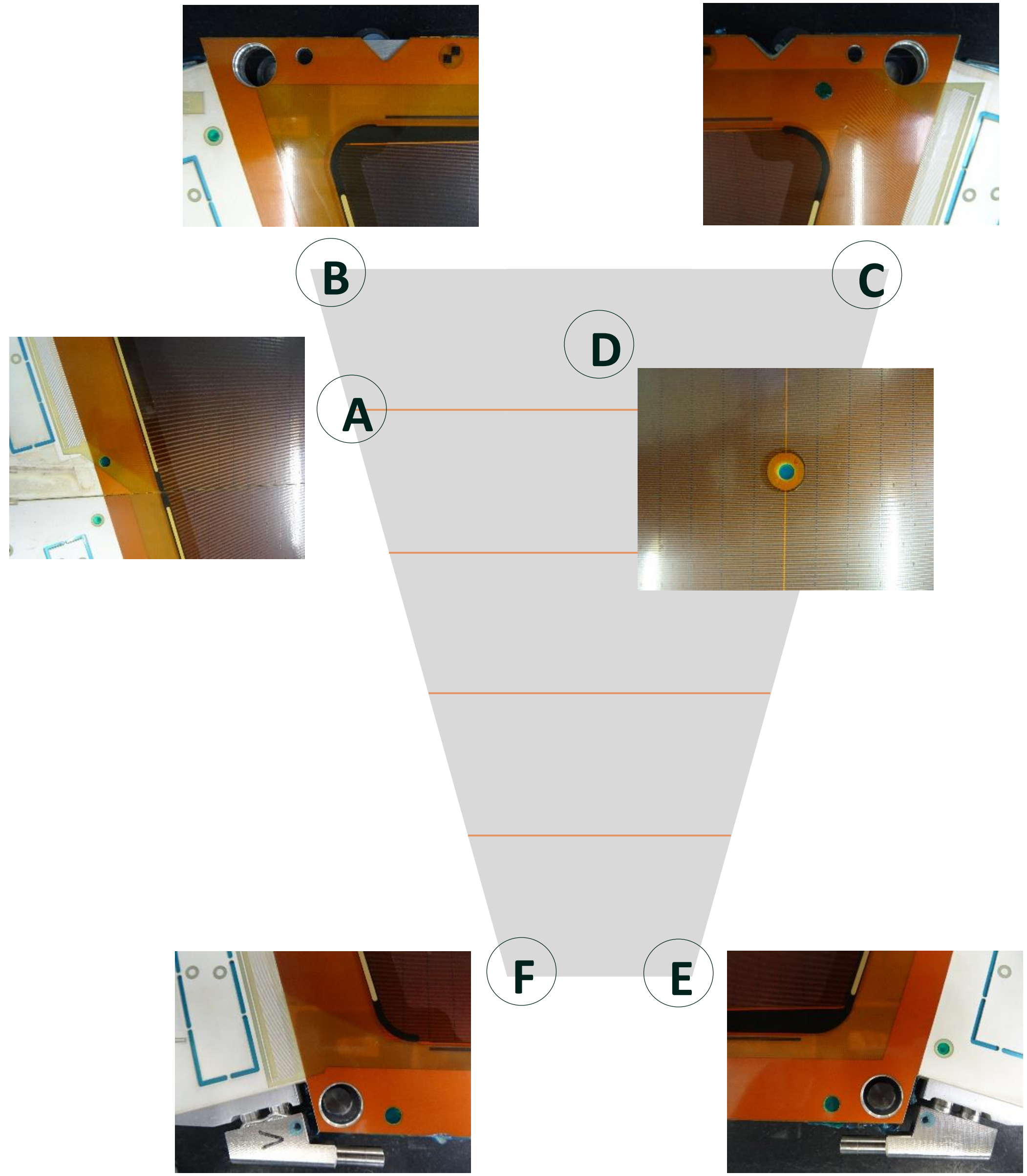
Independently infer for each photo

❸

Fuse 6 scores for final decision

This example session

S23_BonPads



WORKFLOW

Jan '20		Select NN architecture
Feb '20	Local	Collect & prepare input data
Mar '20	Tensorflow, Local or Google Colab	Train network
May '20	Local	Analyse
Jun '20	TfLite inference in gantry control program	Integrate on target
Jul '20		Test
		Deploy

Tune hyperparameters (batch size, learning rate, etc.)

Class balance, data augmentation

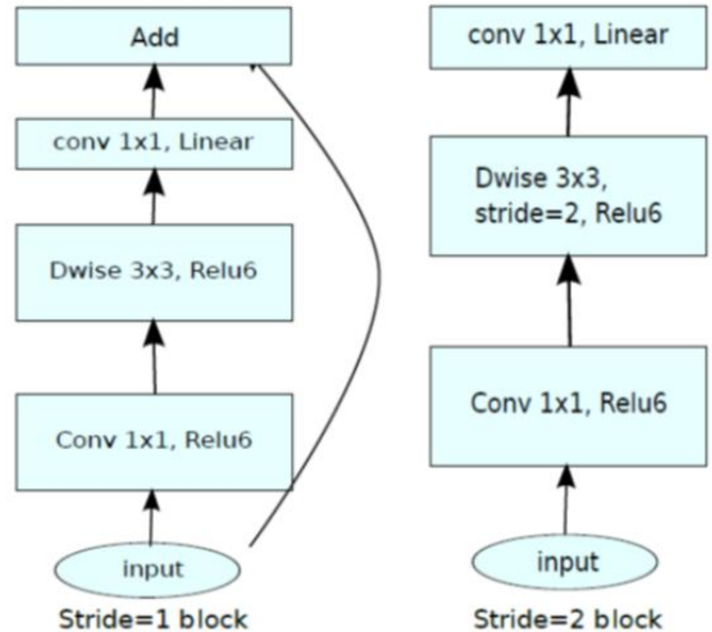
Python / opencv numpy pandas seaborn

C++ / Qt opencv (Windows)

SELECTED NN ARCHITECTURE: MOBILENETV2

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

MobileNetV2 Overall Architecture



MobileNetV2

where t: expansion factor, c: number of output channels, n: repeating number, s: stride. 3×3 kernels are used for spatial convolution.

MobileNetV2 (11 layers)

MobileNetV2 bypass & add layers

MobileNetV2: Inverted Residuals and Linear Bottlenecks, arXiv:1801.04381

TRANSFER LEARNING (1/2)

1- Load a headless pre-trained model from an open depository

```
import tensorflow as tf
import tensorflow_hub as hub
IMG_SIZE = 224 # Specify height and width of image to match the input format of the model
CHANNELS = 3 # Keep RGB color channels to match the input format of the model
IMG_SHAPE = (IMG_SIZE, IMG_SIZE, CHANNELS)
…
feature_extractor_url = "https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4"
feature_extractor_layer = hub.KerasLayer(feature_extractor_url, input_shape=IMG_SHAPE)
```

This model was previously trained on the ImageNet database (14 197 122 images in 21841 categories)

2- Add a classification head to the model, it will produce our label scores as outputs

```
model = tf.keras.Sequential([
    feature_extractor_layer,
    layers.Dense(1024, activation='relu', name='hidden_layer'),
    layers.Dense(nmbrOfLabels, activation='sigmoid', name='output')
])
```

2 fully-connected (dense) layers added

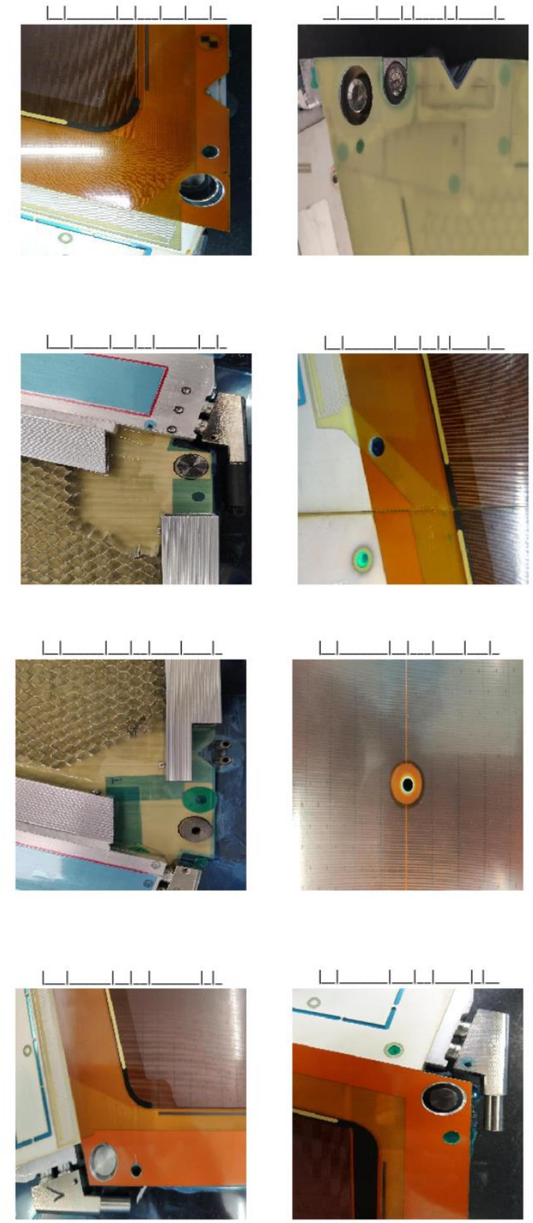
TRANSFER LEARNING (2/2)

3- Model summary

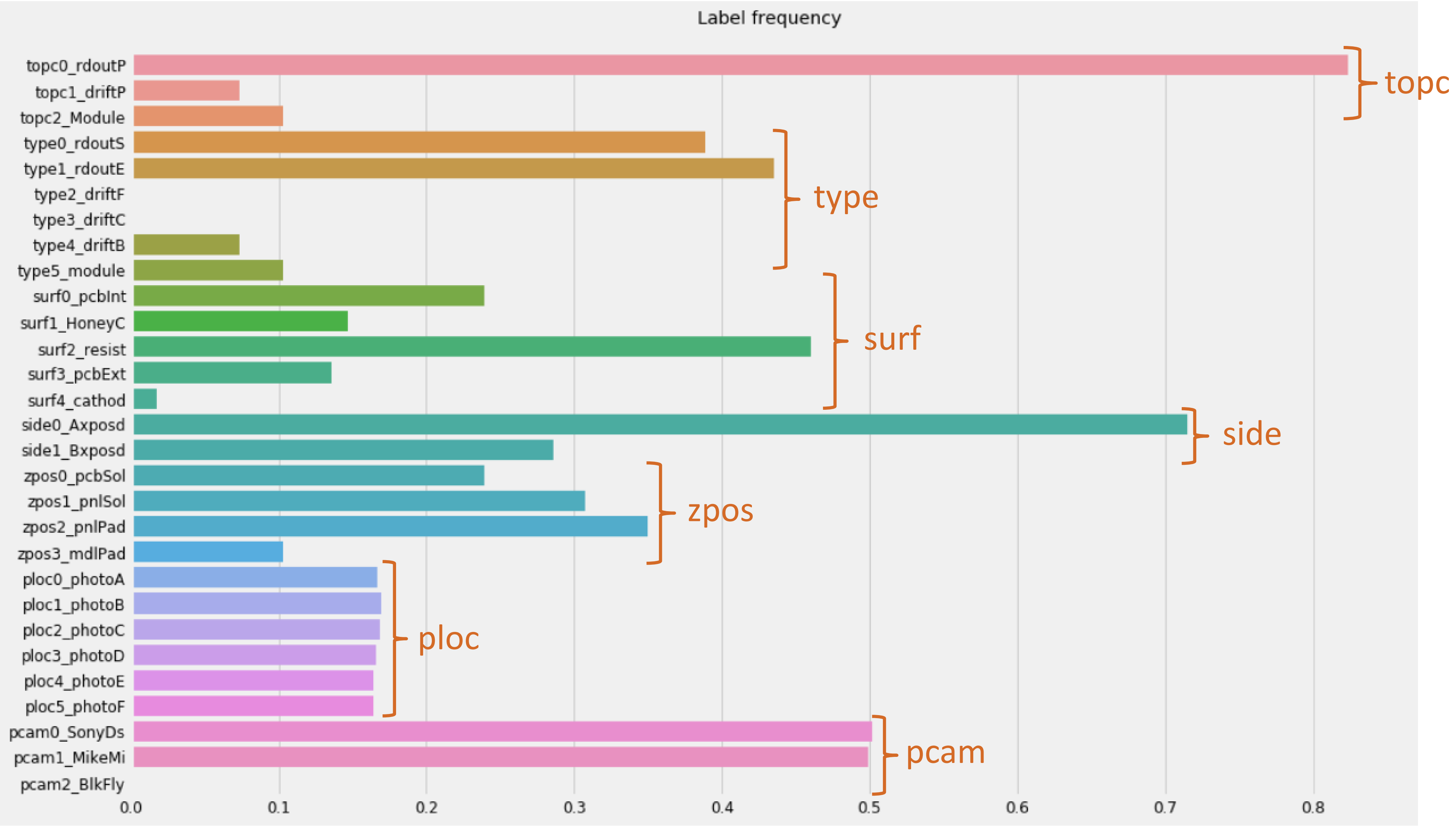
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenetv2_1.00_224 (Model) (None, 7, 7, 1280)        2257984
_________________________________________________________________
hidden_layer (Dense)         (None, 7, 7, 1024)        1311744     = 1024 * (1280 + 1)
_________________________________________________________________
output (Dense)               (None, 7, 7, 29)          29725       = 29 * (1024 + 1)
=================================================================
Total params: 3,599,453
Trainable params: 1,341,469
Non-trainable params: 2,257,984
_________________________________________________________________
```

IMAGE COLLECTION

Session: definition	The set of 6 photos taken at a particular step of prod/QC
60 prod/QC steps observed	Collected Jan ‘20 to Mar ’20, a total of 750 photos
118 resulting sessions	Half taken with smartphone, half with clean room camera
Intrinsic unbalance	Natural sideA/sideB unbalance
Programmatic unbalance	Missing DriftF and DriftC

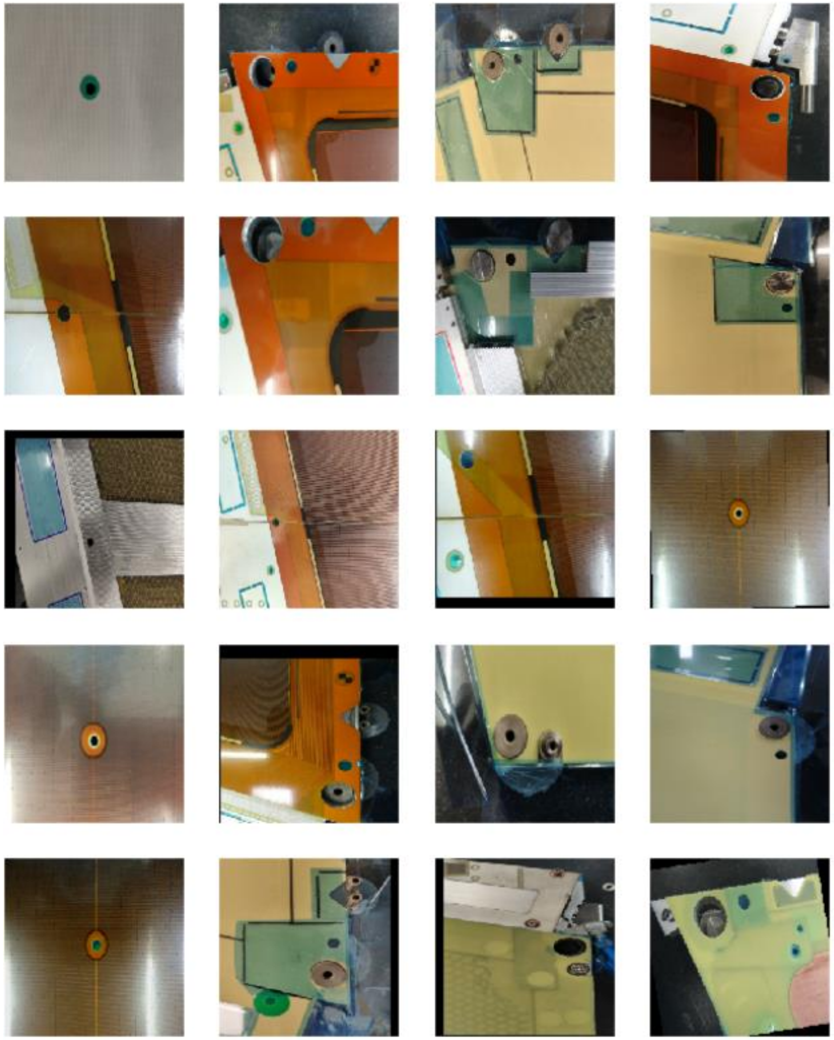


CLASS FREQUENCY & (UN)BALANCE



DATA AUGMENTATION

Balance	Duplicate with data augmentation, equalizing sideA & sideB images
Data augmentation	Final training dataset of 7000+ photos
Random image alteration	Shift, rotate, alter brightness, alter saturation, smooth, crop and scale



Training (1/2): Custom multi-class loss function

Usually, binary cross entropy: negative log likelihood -log(p) of an observation being of a specific class with the model predicting a probability p for that class.

Does not work for correlated multi-class

Use F1-score (harmonic mean of Precision and Recall) instead, F1 = 2.TP / (2.TP + FN + FP)

		True condition				
	Total population	Condition positive	Condition negative	Prevalence = $\frac{\Sigma \text{ Condition positive}}{\Sigma \text{ Total population}}$	Accuracy (ACC) = $\frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$	
Predicted condition	Predicted condition positive	**True positive**	**False positive**, Type I error	Positive predictive value (PPV), Precision = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Predicted condition positive}}$	False discovery rate (FDR) = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Predicted condition positive}}$	
	Predicted condition negative	**False negative**, Type II error	**True negative**	False omission rate (FOR) = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Predicted condition negative}}$	Negative predictive value (NPV) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Predicted condition negative}}$	
		True positive rate (TPR), Recall, Sensitivity, probability of detection, Power = $\frac{\Sigma \text{ True positive}}{\Sigma \text{ Condition positive}}$	False positive rate (FPR), Fall-out, probability of false alarm = $\frac{\Sigma \text{ False positive}}{\Sigma \text{ Condition negative}}$	Positive likelihood ratio (LR+) = $\frac{\text{TPR}}{\text{FPR}}$	Diagnostic odds ratio (DOR) = $\frac{\text{LR+}}{\text{LR}-}$	F_1 score = $2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$
		False negative rate (FNR), Miss rate = $\frac{\Sigma \text{ False negative}}{\Sigma \text{ Condition positive}}$	Specificity (SPC), Selectivity, True negative rate (TNR) = $\frac{\Sigma \text{ True negative}}{\Sigma \text{ Condition negative}}$	Negative likelihood ratio (LR−) = $\frac{\text{FNR}}{\text{TNR}}$		

To maximise F1-Score, minimize 1 – F1-score

F1-Score is not differentiable (discrete), use soft-F1 instead (continuous probabilities)

Average soft-F1 accross all category labels -> macro-soft-F1

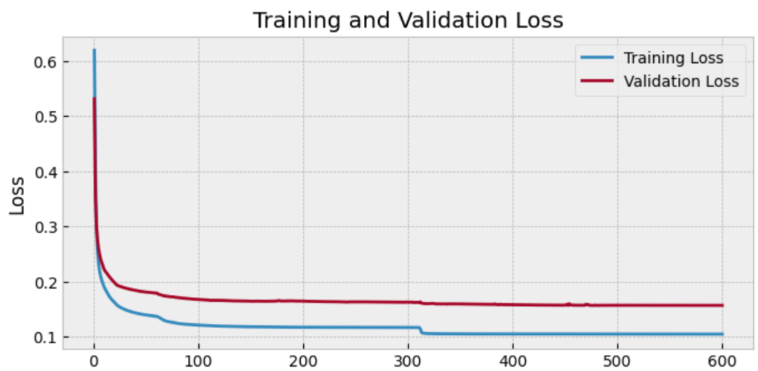
TRAINING (2/2) : FIT & VALIDATE THE MODEL

```
Epoch 599/600
30/30 [==============================] - 6s 190ms/step - loss: 0.1044 - macro_f1: 0.8956 - val_loss: 0.1565 - val_macro_f1: 0.8448
Epoch 600/600
30/30 [==============================] - 6s 190ms/step - loss: 0.1044 - macro_f1: 0.8956 - val_loss: 0.1565 - val_macro_f1: 0.8448

Initial training took 0h:57m:19s
```

alize the learning curves on the training and validation sets when using the macro soft-F1 loss.
function that plots learning curves was implemented and imported from `utils` module.

```
losses, val_losses, macro_f1s, val_macro_f1s = learning_curves(history)
```

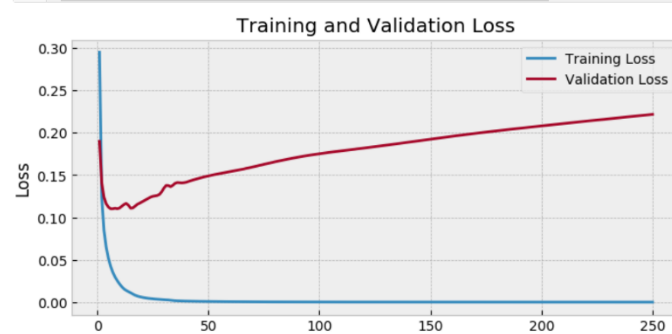


Training and Validation Macro F1-score

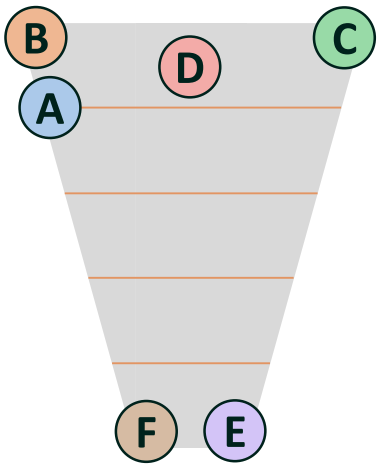
Optimizing for Macro-F1 score converges …

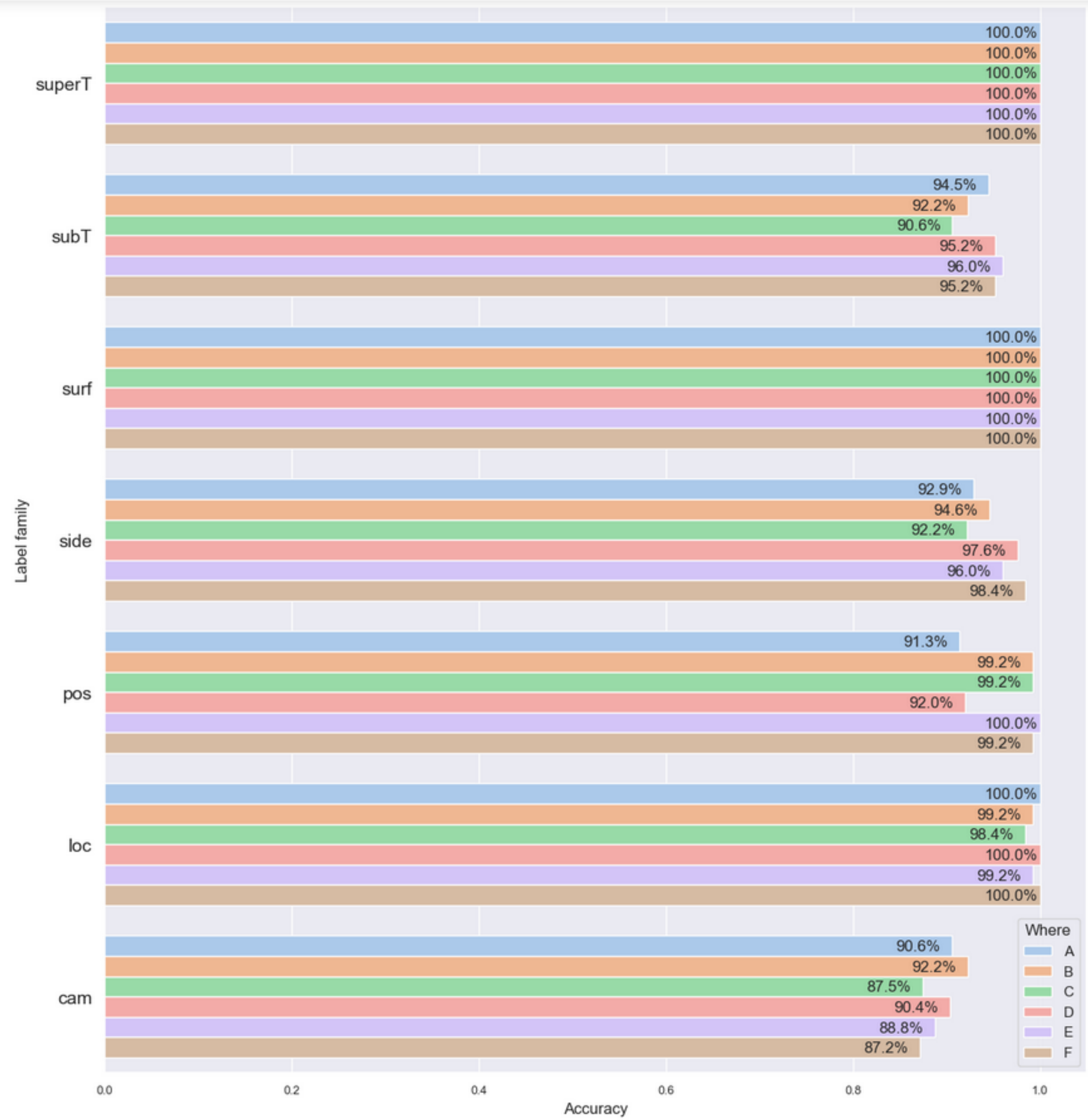
```
1 model_bce_losses, model_bce_val_losses, model_bce_macro_f1s, model_bce_val_macro_f1s
```

… where binary cross entropy loss fails



ANALYSIS: PER-IMAGE ACCURACY, FOR EACH LOCATION





ANALYSIS: PER-IMAGE ACCURACY, FOR EACH SESSION TYPE

AiPcb

BonPads

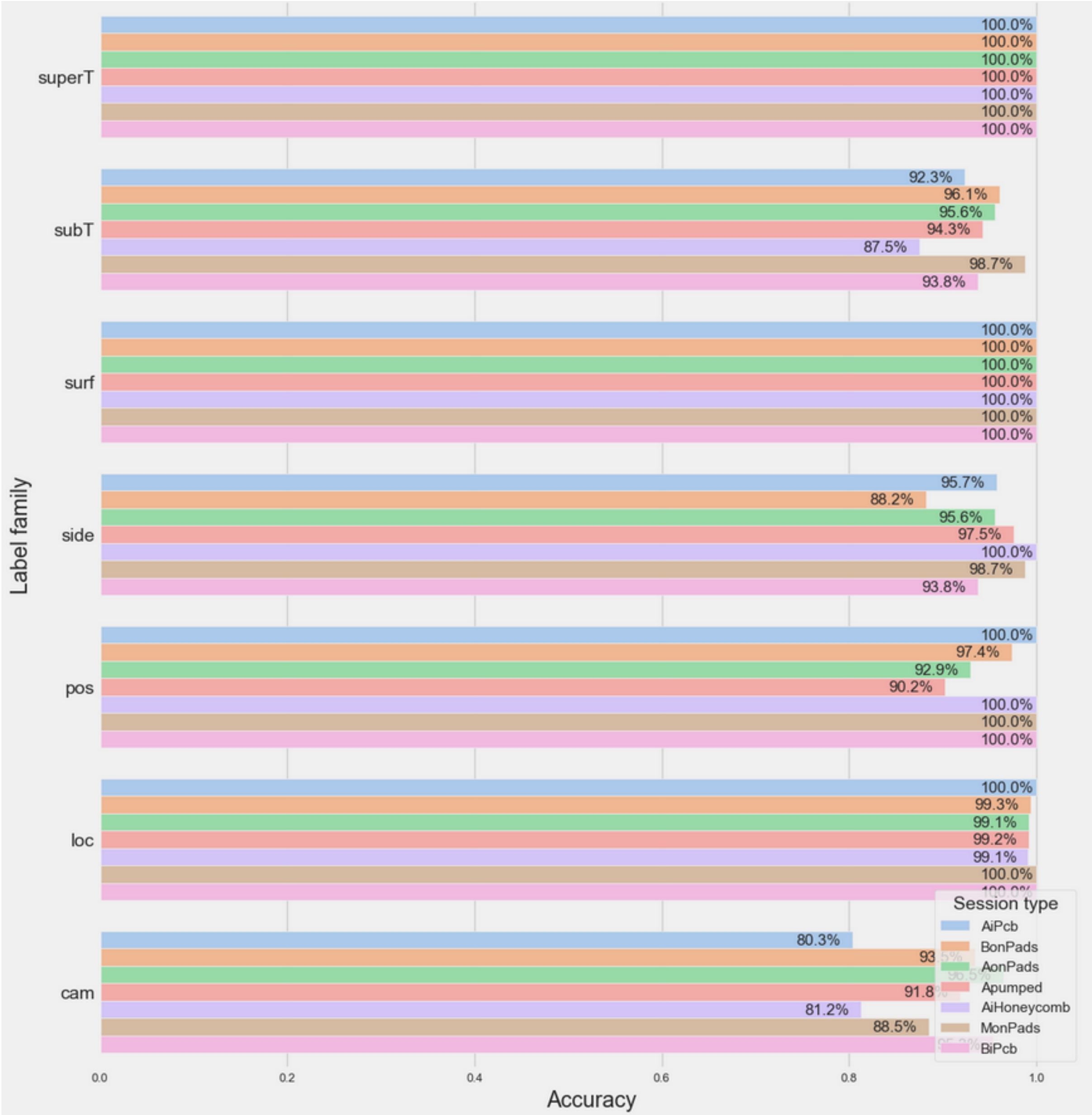
AonPads

Apumped

AiHoneycomb

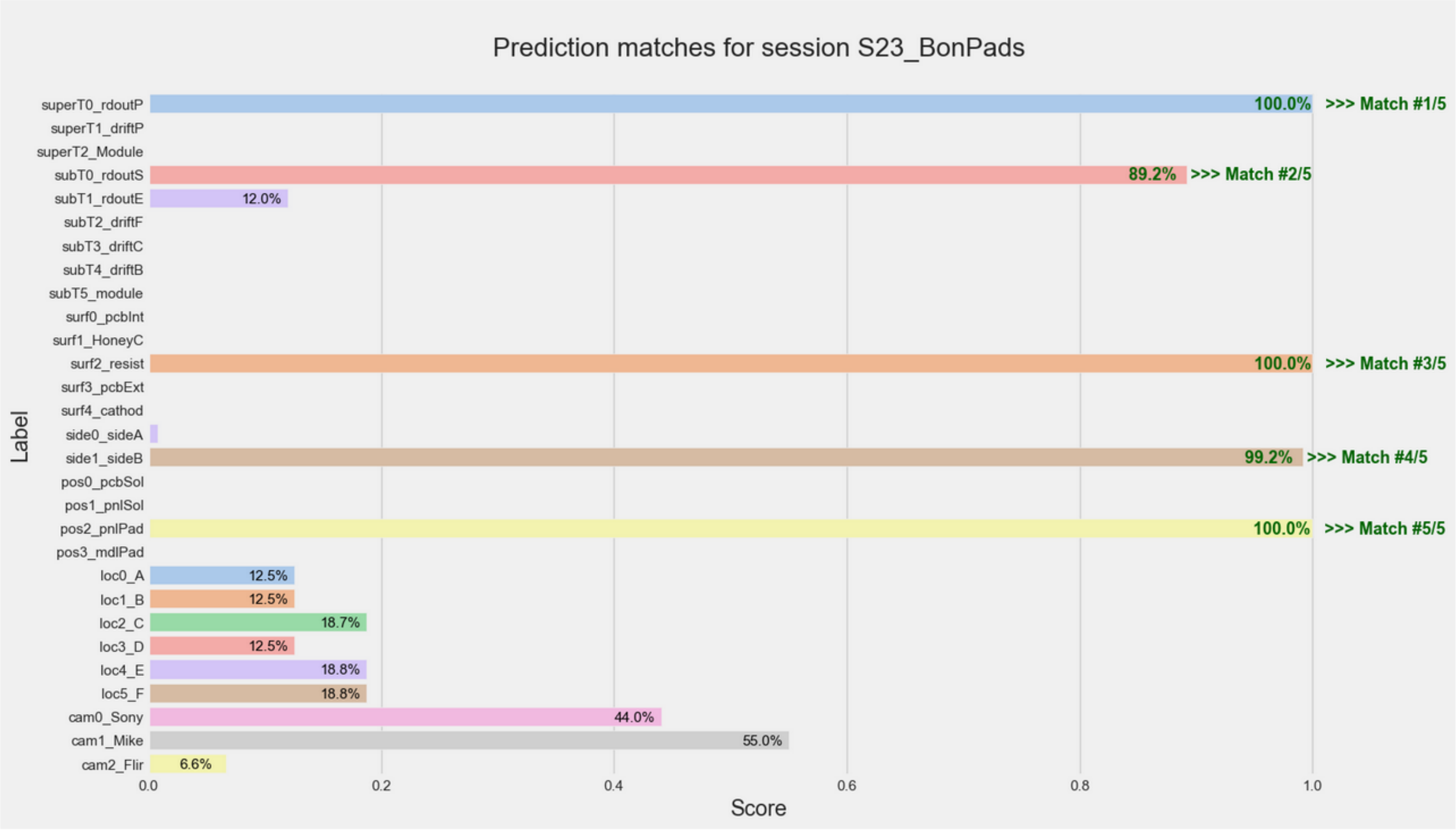
MonPads

BiPcb

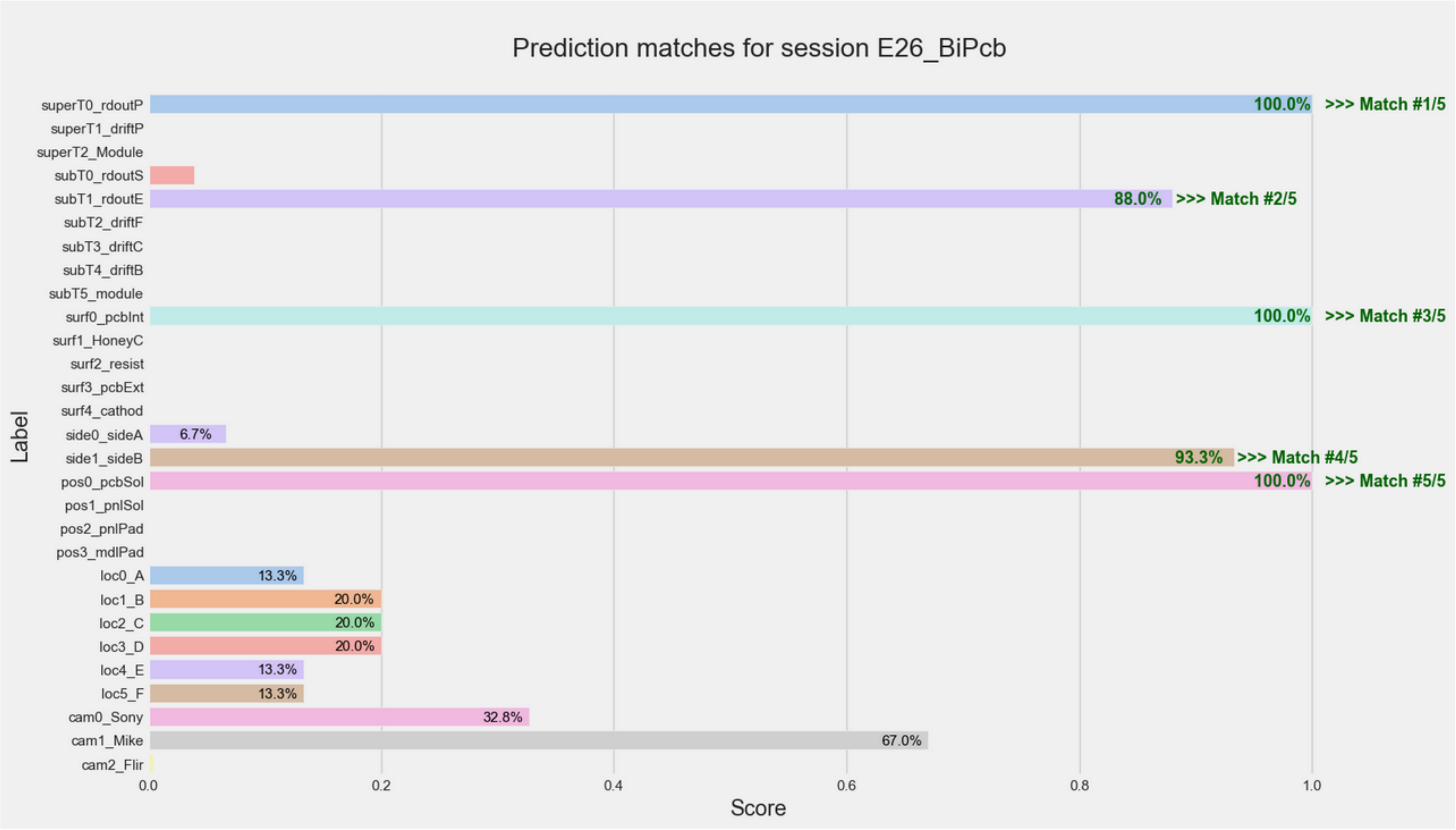


Fusion analysis: Session scores (1/2)

```
Session #45: S23_BonPads, sideA score divided by 1.0, then mean of A to F scores
!!!!!!!!!!!! Quine #45/45, 5/5 prediction families have matched... !!!!!!!!!!!!!!
```



```
Session #60: E26_BiPcb, sideA score divided by 1.0, then mean of A to F scores
!!!!!!!!!!!! Quine #60/60, 5/5 prediction families have matched... !!!!!!!!!!!!!
```




```
!!!!!!!!!!!!!! Bingo, all 60 sessions have matched for all prediction families...
```

Fine, but will it generalize correctly?

Save frozen model

Export trained model with frozen graph and constant weights

```
Convert the model to a TFLite model, direct
converter = tf.lite.TFLiteConverter.from_saved_model(newestSoftF1dir)
tflite_model = converter.convert()
open("reloadedF1.tflite", "wb").write(tflite_model)
```

Install tensorflowLite

C++ Library, for inference only

```
// Load model, build ther interpreter, allocate tensor buffers
model_ = tflite::FlatBufferModel::BuildFromFile("../NswFiles/Rainfrog/Model/reloadedF1.tflite");
InterpreterBuilder builder(model(), resolver);
builder(&interpreter);
TFLITE_MINIMAL_CHECK(interpreter->AllocateTensors() == kTfLiteOk);
```

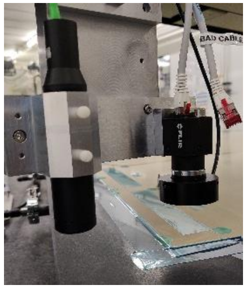
Run inference

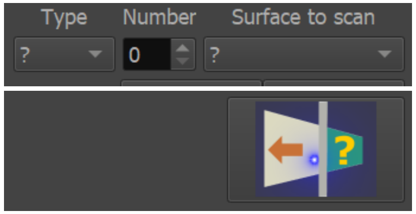
```
// Feed network, run inference and save ouputs
// Image data must first be formatted to fit network input (rgb order, float)
uint8_t* imgPixelPtr = inputImage.ptr<uint8_t>(0);
float* feedIn = interpreter->typed_tensor<float>(input);
prepareTfliteFloatInput(imgPixelPtr, feedIn);
TFLITE_MINIMAL_CHECK(interpreter->Invoke() == kTfLiteOk);
for (int i = 0; i < labelNames.size(); ++i) imageScore[i] = interpreter->typed_output_tensor<float>(0)[i];
```

Integrate to existing control program

Images taken by Flir camera on gantry head, requires 4-photo stitching

Unchanged GUI, Rainfrog is launched when user selects « ? » scan command

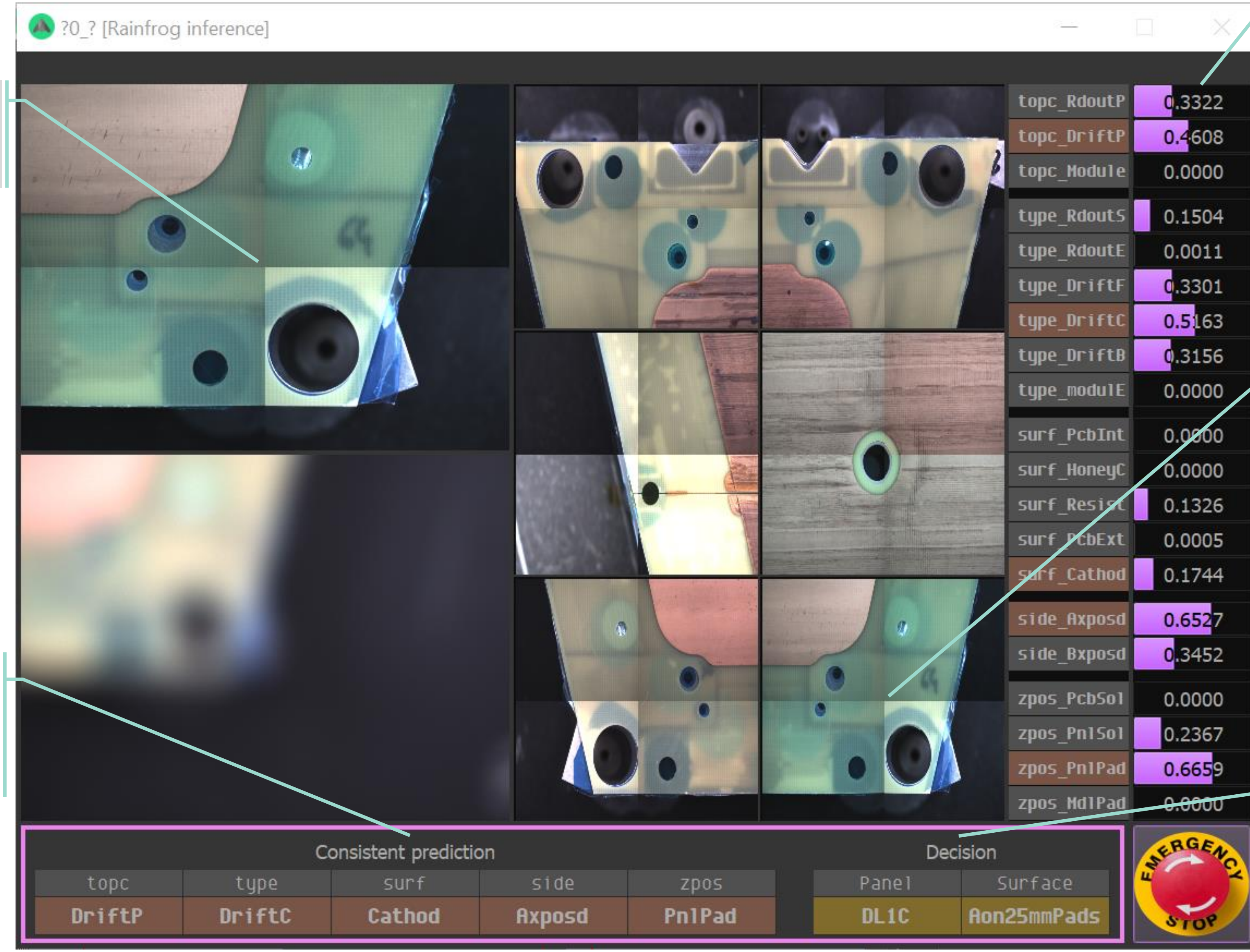




FIELD TEST (1/4): 1ST «CORRECT» INFERENCE: B32_AONPADS (LOUISE)

Hi res 4-photo assembly

5-Family prediction after fusion

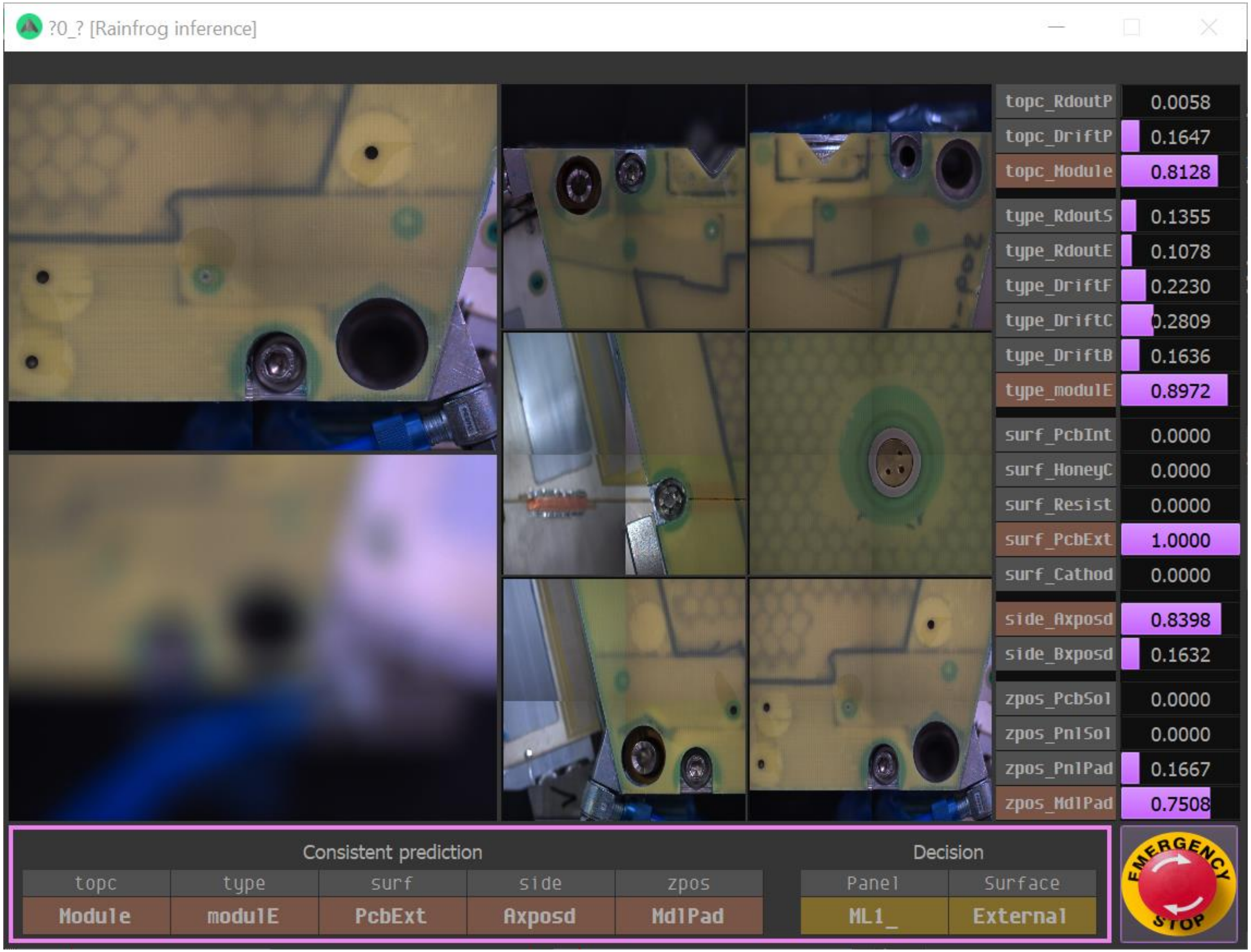


Category scores after fusion

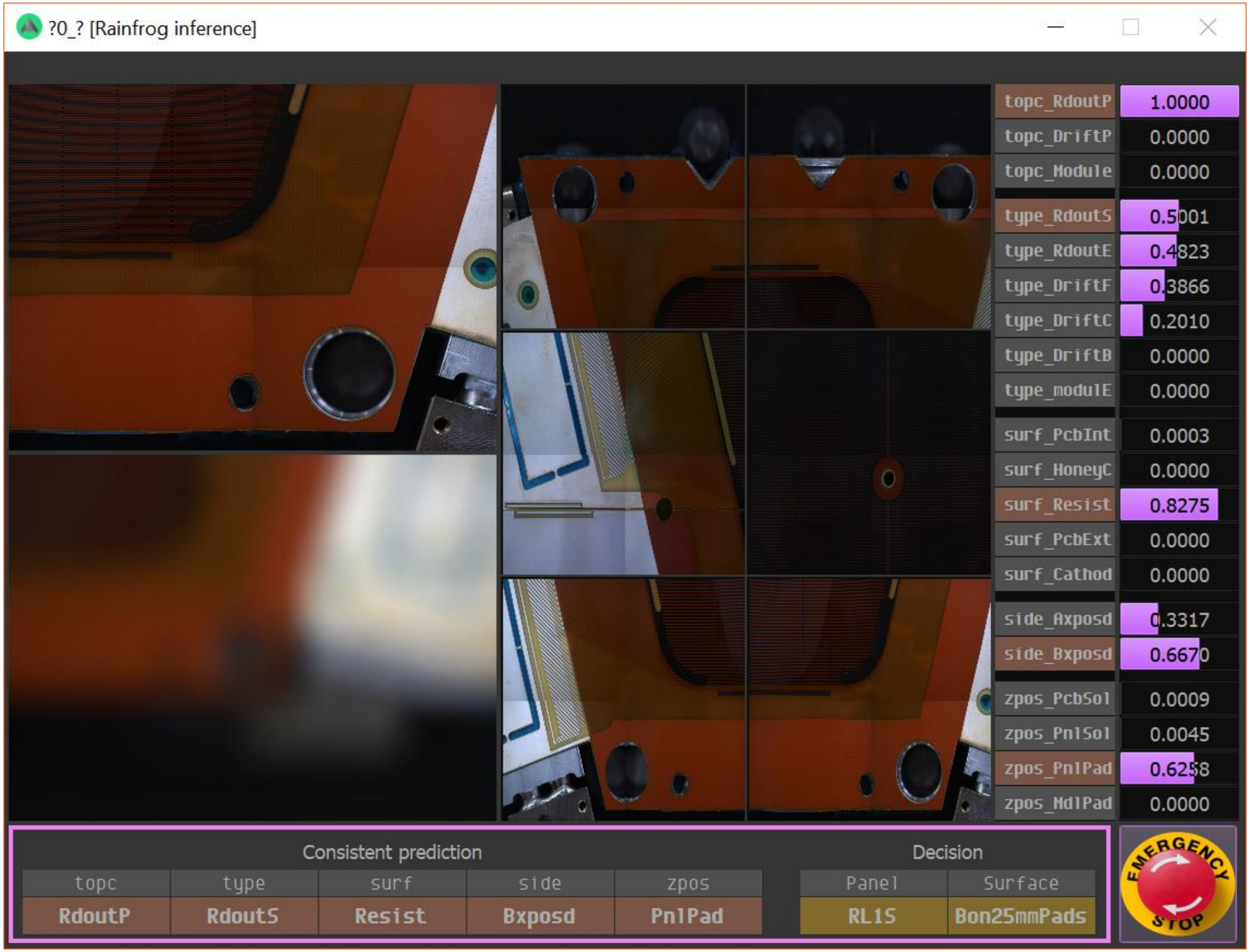
Low res 224*224 image used for inference @loc E

Final decision

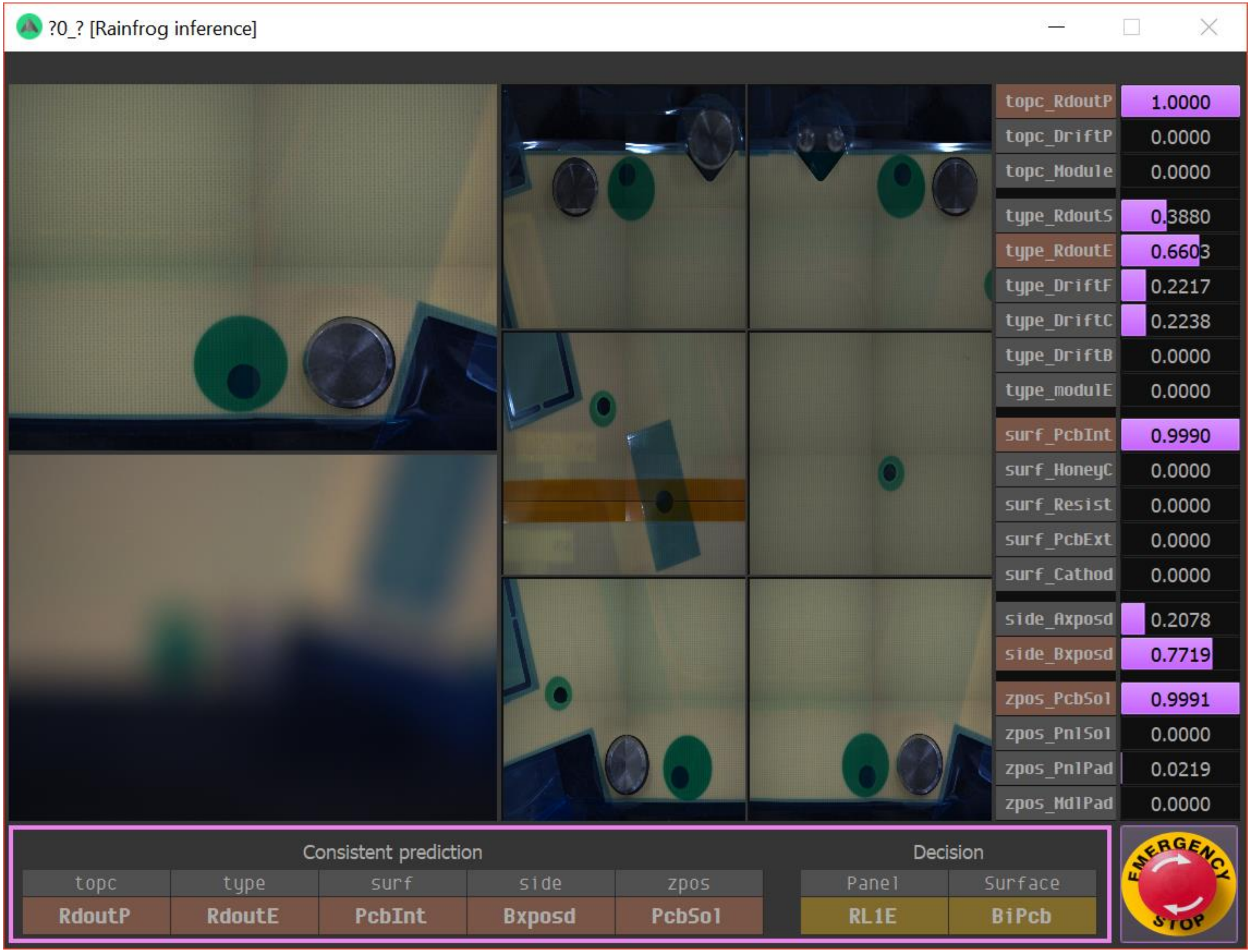
FIELD TEST (2/4): 2^{D} CORRECT INFERENCE, M22_MONPADS (THELMA)



FIELD TEST (3/4): WRONG PREDICTION, E28_BONPADS (THELMA)



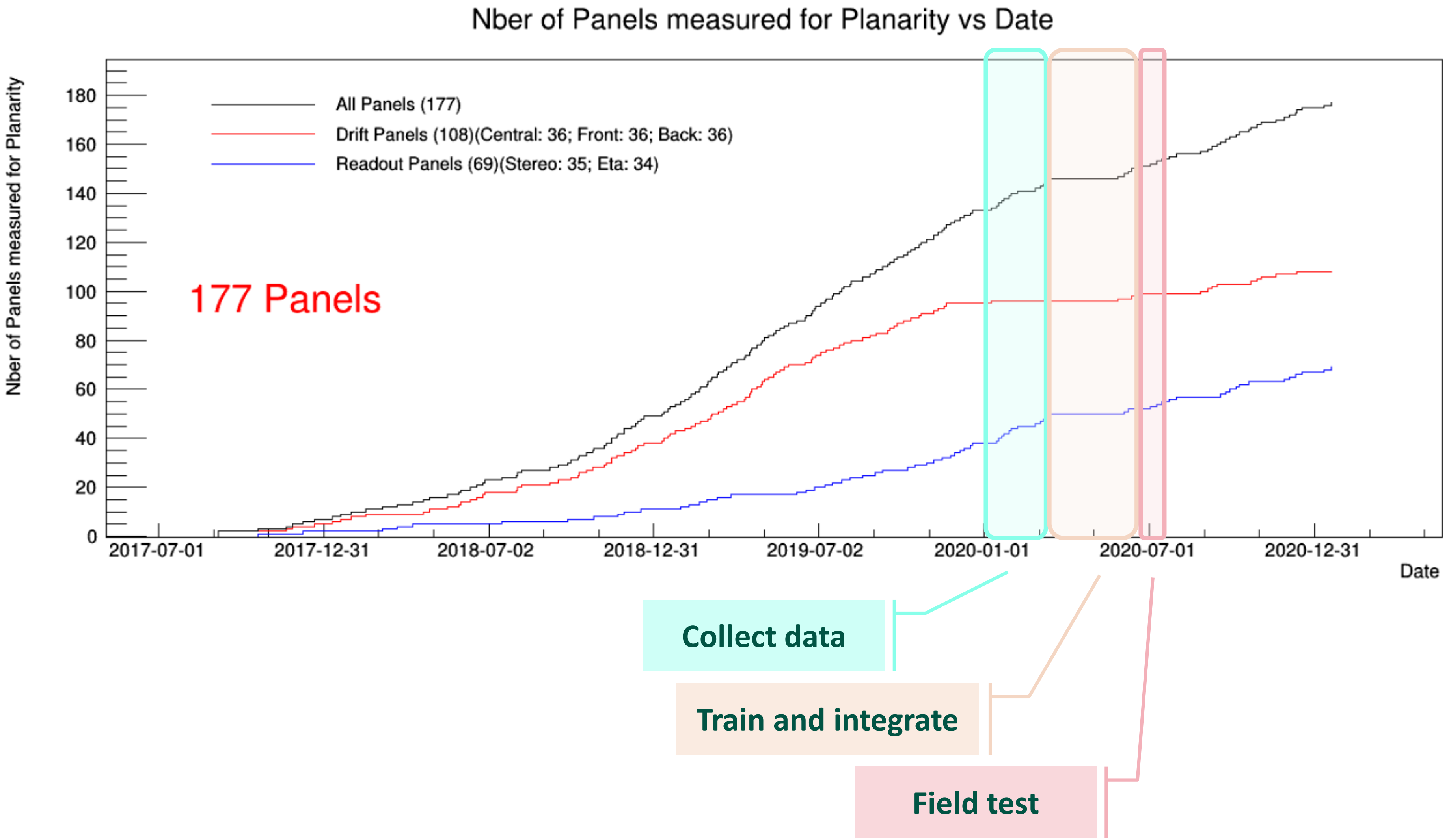
FIELD TEST (4/4): CORRECT PREDICTION, E29_BIPCB (THELMA)



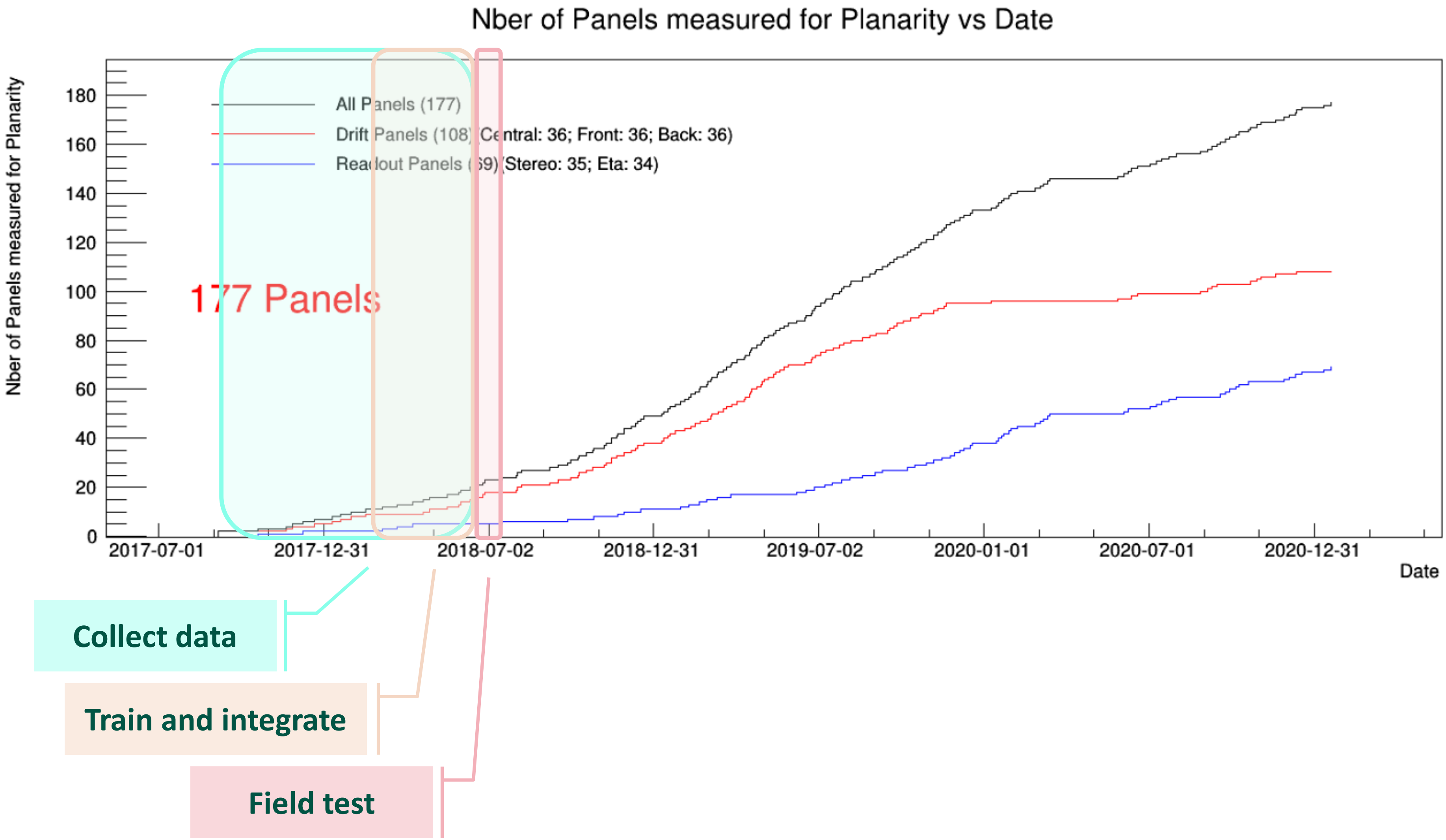
SUMMARY

Accuracy	Accuracy found to be ~90% or more for individual patch images on initial dataset
	After patch fusion, achieved 100% on initial dataset
Integration issues	Run-time camera field of view was different from the fov of initial training images
	Automatic vertical sensing by contrast analysis was difficult to tune for all surface types
Possible improvements	Better NN architecture : MobileNetV3?
	Add fine tuning phase to training
☹ Well, was too late for deployment…	Such applications need to be carefully scheduled ahead of production
	In July '20, 147 panels had been measured so far / 177 total (32 + some spare modules built)

Production progress as of Feb. '21



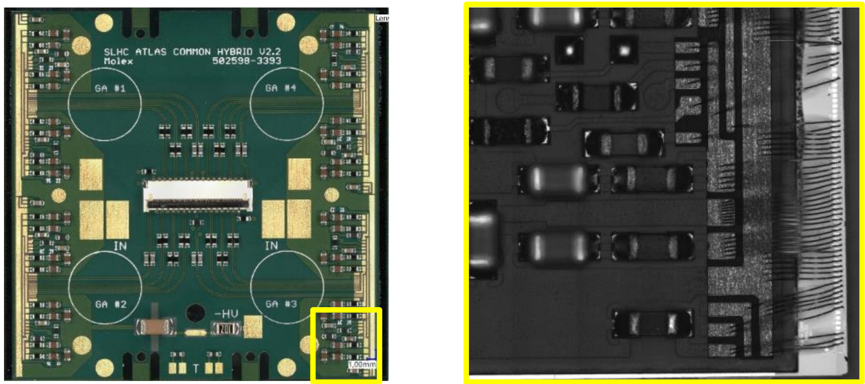
SHOULD HAVE BEEN SCHEDULED AHEAD OF PRODUCTION…



Other ML applications for test automation?

Atlas ITK

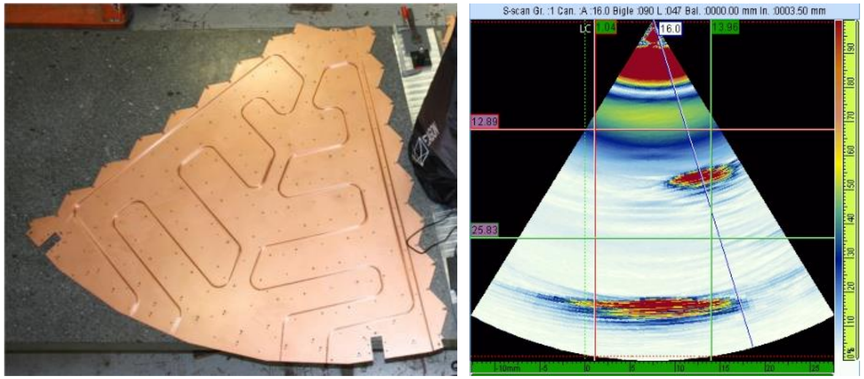
Verification of bonding by analysis of patch images



Atlas ITK 4*4 cm^2 module.
Closeup on bottom right area to show a bonding section.

CMS HGCal

Verification of pipe soldering by ultrasound image analysis



CMS HGCal copper plates with cooling pipes.
Illustration of ultrasound imaging of soldering sections.

Both address the challenging issue of anomaly detection in an early phase

Thanks …