Canny background replacement and training

Marco Ferri - 30th September 2020

The purpose of this report is to show how the original Dario's model performs out of the drone arena. Firstly, an algorithm for background removal is proposed, then some training experiments with various backgrounds are presented. Finally, we find a model which is almost as powerful on new backgrounds as the original model is inside the arena drone.

Source code is available here.

Background removal

The algorithm which performs background removal or replacement is based on Canny Edge Detection for finding the edges of the image and so the contours. These are used for creating a mask around the subject in the image, for isolating it from the background. A core aspect of the Canny function is the correct choice of the parameters `minVal` and `maxVal`, since it is essential for distinguishing the subject edges from the rest and later keeping the right contours.
Background removal is critical for background replacement, used in the next sections.

Several experiments have been made with different parameters for optimizing the background removal, but no combination of `minVal` and `maxVal` is optimal with respect to the given dataset. Some scenes would require more filtering, while others may disappear completely if strong filtering is chosen; only deep human dataset inspection can reveal the lead to the background removal, which may require a lot of time. Up to now, the whole dataset has been filtered with the same `minVal` and `maxVal` parameters, respectively set to 100 and 400.

Example below shows how background removal performed on a subset of the dataset. Most of the time the person is well detected, while other times it completely disappears or even results in a fatal error (red frames, ignored later). Room baseboard (line between the floor and the wall) is often still present in the image, but it becomes almost invisible when merged with replacement backgrounds.

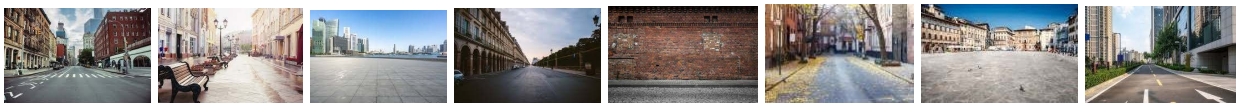
Overview on different types of backgrounds

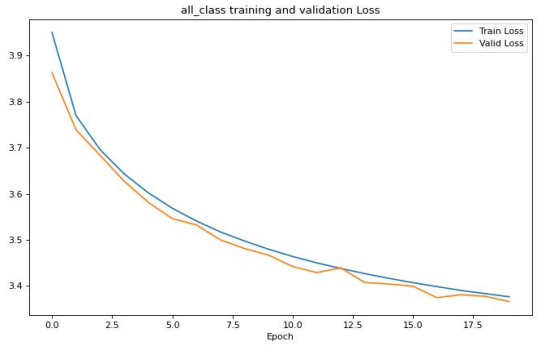
This section aims to compare different types of backgrounds (street, room, park) that have been chosen for replacing the drone arena background. All the tests are carried out on a 50k samples dataset, sampled from the original dataset by skipping 5 frames at a time and replacing the background with other 8 different backgrounds.

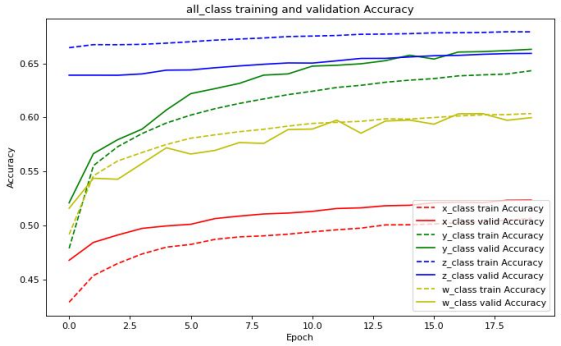
The charts below show the performance, keeping the original model weights, so that just the final classification layers are learned. Rooms and street backgrounds seem to obtain the best performance, while parks are quite hard to handle because of the intrinsic variety of such environments.

We use the **`categorical_crossentropy` loss** and standard **accuracy** for a multi-class problem on a multi-output model (each stands for a single variable x, y, z or w).

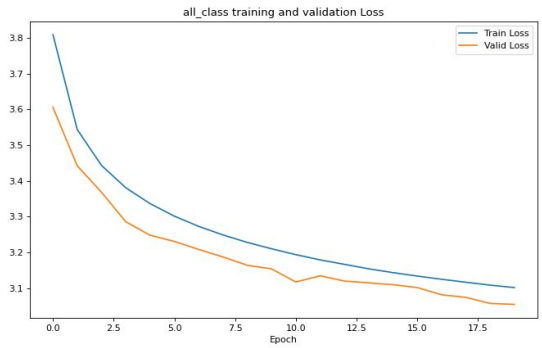
Original weights, 8 Street backgrounds, 50k samples (skip 5)

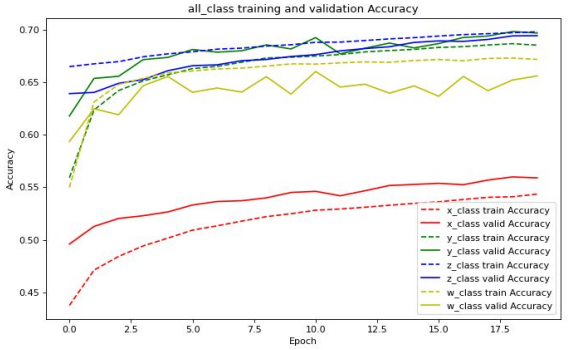




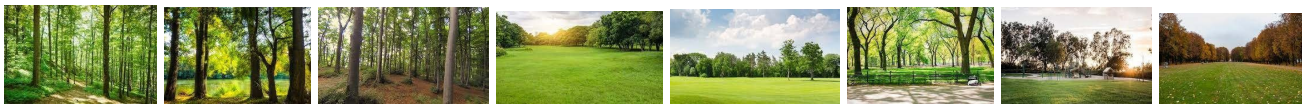


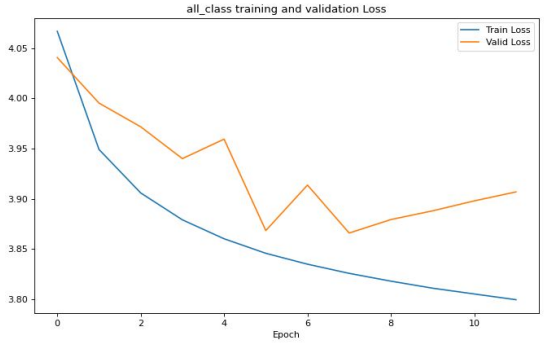
Original weights, 8 Rooms backgrounds, 50k samples (skip 5)

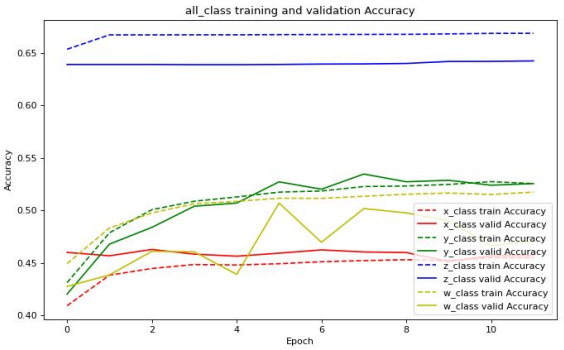




Original weights, 8 Parks backgrounds, 50k samples (skip 5)

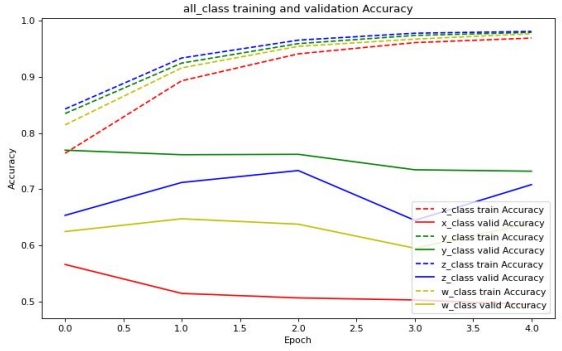






Retrain All, 8 Rooms backgrounds, 50k samples (skip 5)

Running the same model of before but with all trainable layers, results in a huge overfit.

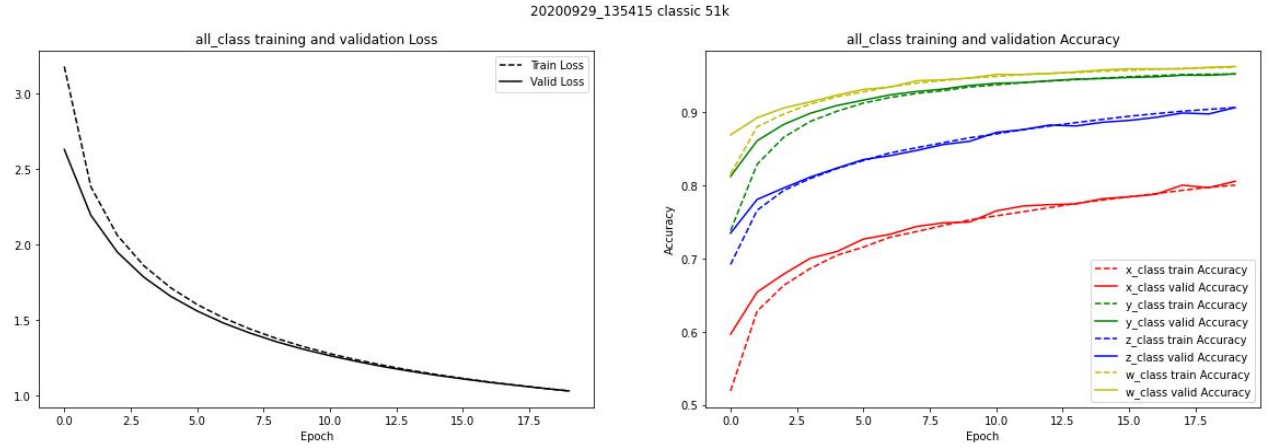


Training experiments

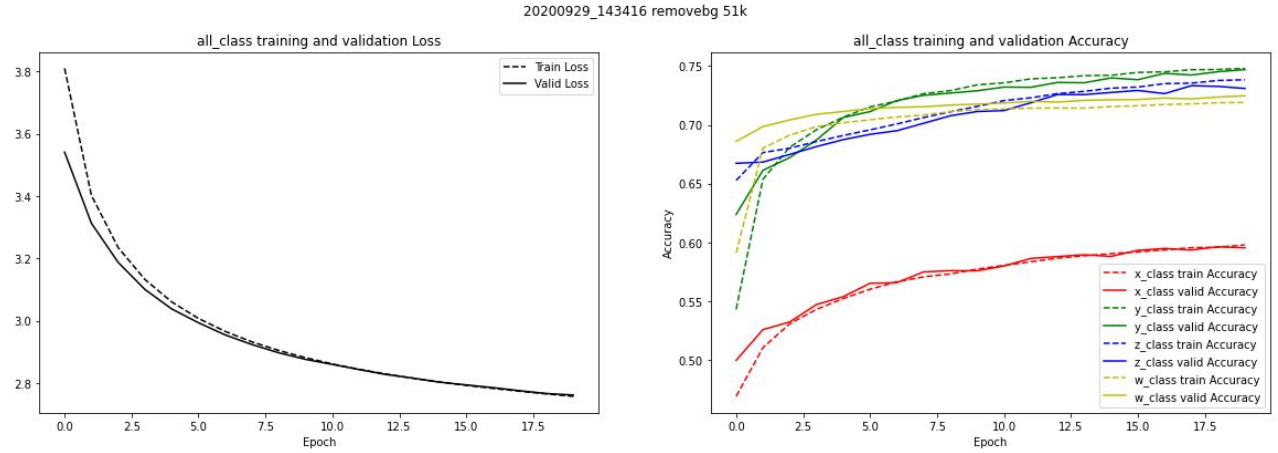
Original model weights on Arena, Blank and Replaced backgrounds

We end up having three types of dataset: original arena, blank and replaced backgrounds.
Each of them requires performance inspection, which follows.

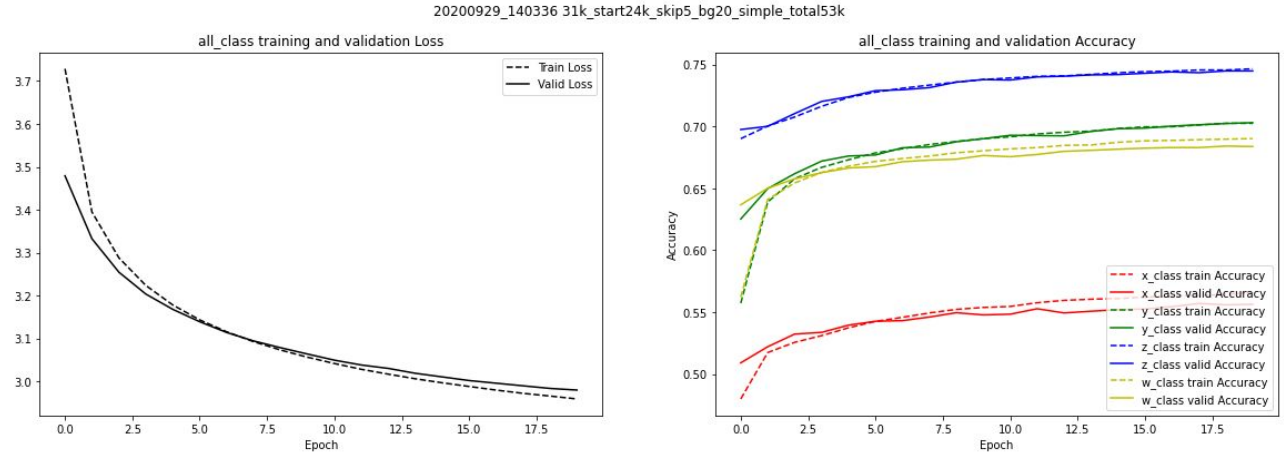
The first chart shows how the **original model** performs on the original dataset. Loss is equal 1.



The second chart shows how the original model performs on **blank background** dataset. Although the trend seems the same, the loss improvement stops at 2.8 after the 20th epoch.

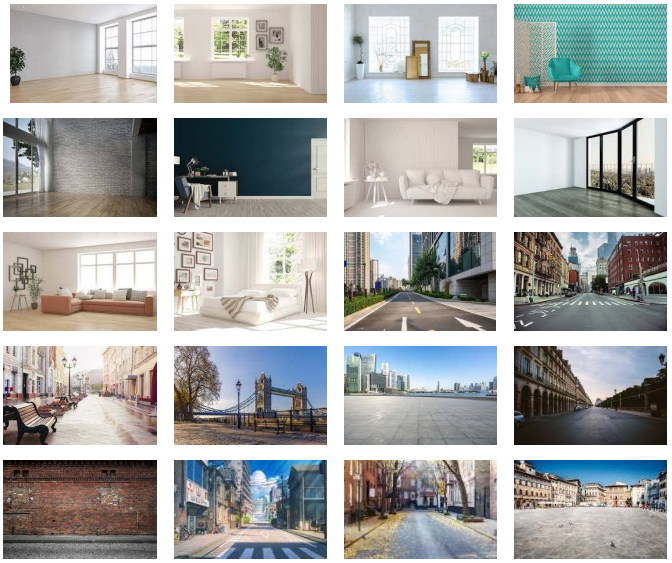


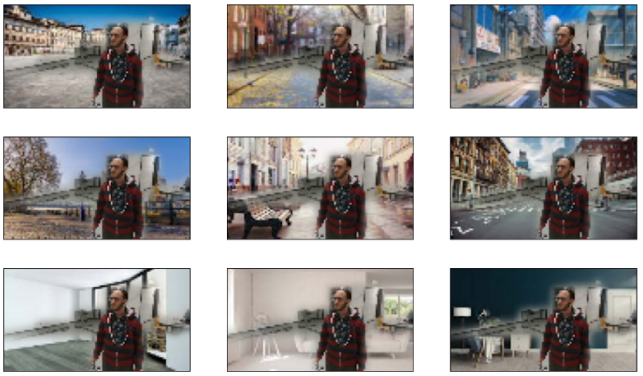
Finally, the dataset with **replaced backgrounds**, in which we are interested in, obviously achieves the worst performance (still with the original model) due to its complexity, but with a loss value similar to the previous blank dataset.



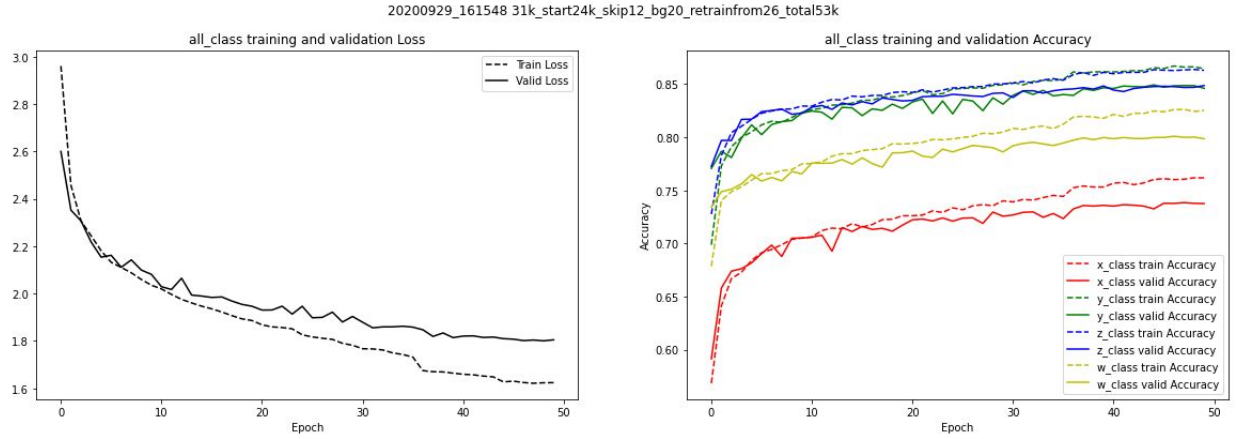
Different retraining layers comparison

Here we show how the number of trainable layers affect the performance of the model, taking into consideration a dataset composed of **20 different street and room backgrounds** (listed below). The charts consider **three different models**, respectively trainable from the 26th (`batch_normalization_7`), 24th (`conv2d_9`) and 22nd (`activation_6`) layers with respect to the following architecture.

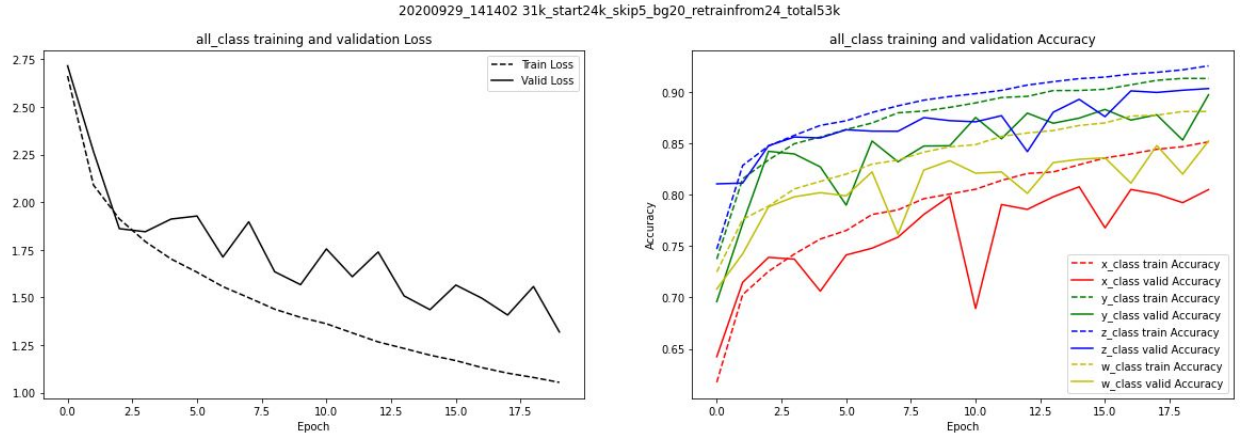




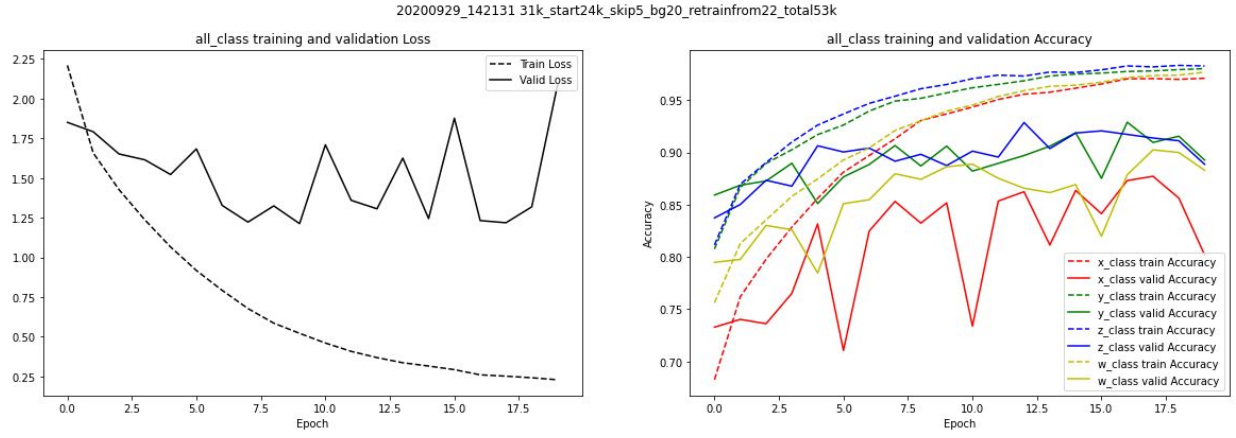
Trainable from **layer 26**. Train and validation losses (`categorical_crossentropy`) start together, but the gap increases after 20 epochs, with a validation loss of about **1.8** at the 50th epoch.



Trainable from **layer 24**. Train and validation losses separate immediately but they decrease until the validation loss reaches the value of about **1.3** at the 20th epoch.

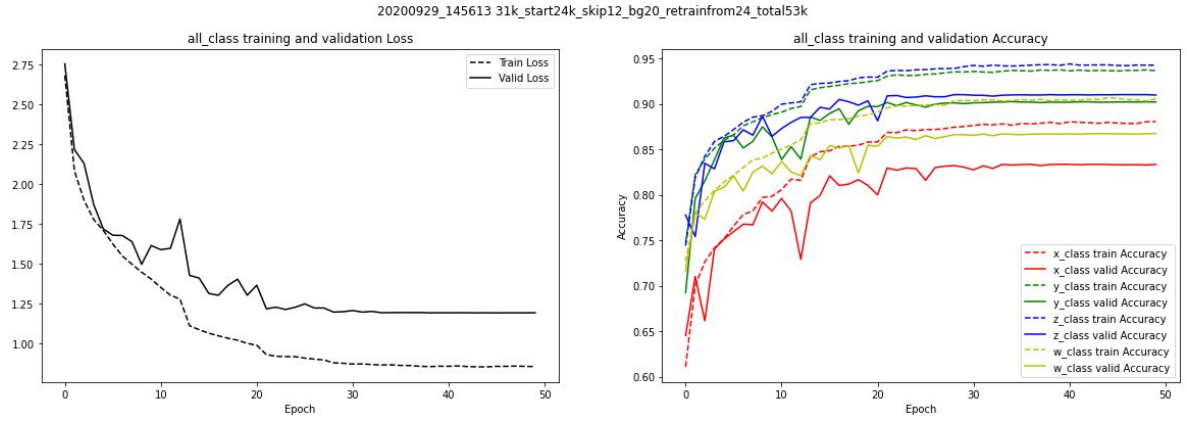


Trainable from **layer 22**. The model clearly **overfits** after 8 epochs.



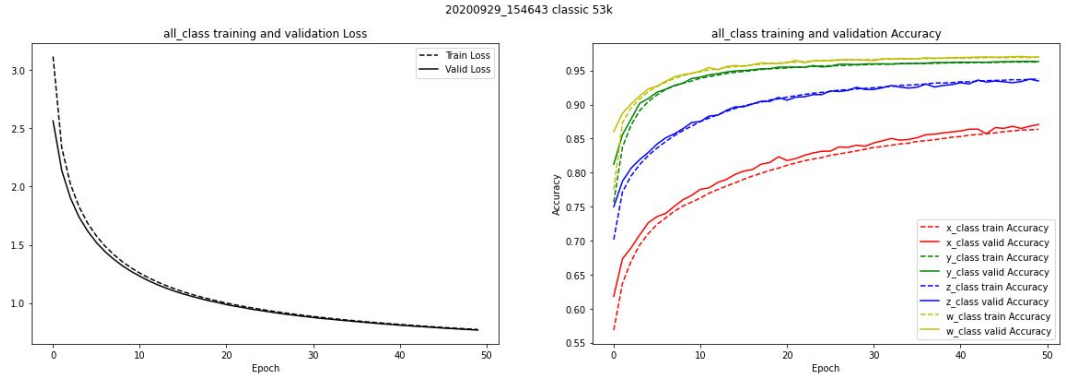
Final choice comparison with Original dataset performance

As it resulted to be the best option, here we show a training session with a **model trainable from the layer 24**, using a **53k samples** dataset composed of **20 backgrounds** (streets, rooms).




```
train_loss: 0.8541
x_loss: 0.2982, y_loss: 0.1601, z_loss: 0.1577, w_loss: 0.2381
x_accuracy: 0.8805, y_accuracy: 0.9365, z_accuracy: 0.9425, w_accuracy: 0.9057
val_loss: 1.1917
val_x_loss: 0.3919, val_y_loss: 0.2394, val_z_loss: 0.2355, val_w_loss: 0.3249
val_x_accur: 0.8335, val_y_accur: 0.9020, val_z_accur: 0.9098, val_w_accur: 0.8675
```

... we compare it with the drone arena original same-size dataset and original model (no retrain):




```
train_loss: 0.7719
x_loss: 0.3841 - y_loss: 0.1109 - z_loss: 0.1805 - w_loss: 0.0963
x_accuracy: 0.8637 - y_accuracy: 0.9623 - z_accuracy: 0.9377 - w_accuracy: 0.9696
val_loss: 0.7668
val_x_loss: 0.3759 - val_y_loss: 0.1115 - val_z_loss: 0.1829 - val_w_class_loss: 0.0965
val_x_accur: 0.8708, val_y_accur: 0.9632, val_z_accur: 0.9347, val_w_accur: 0.9700
```

The performance of the new model is worse than the original, but still comparable. It may be that using the selected model to fly the drone out of the arena turns out to be a good flight.

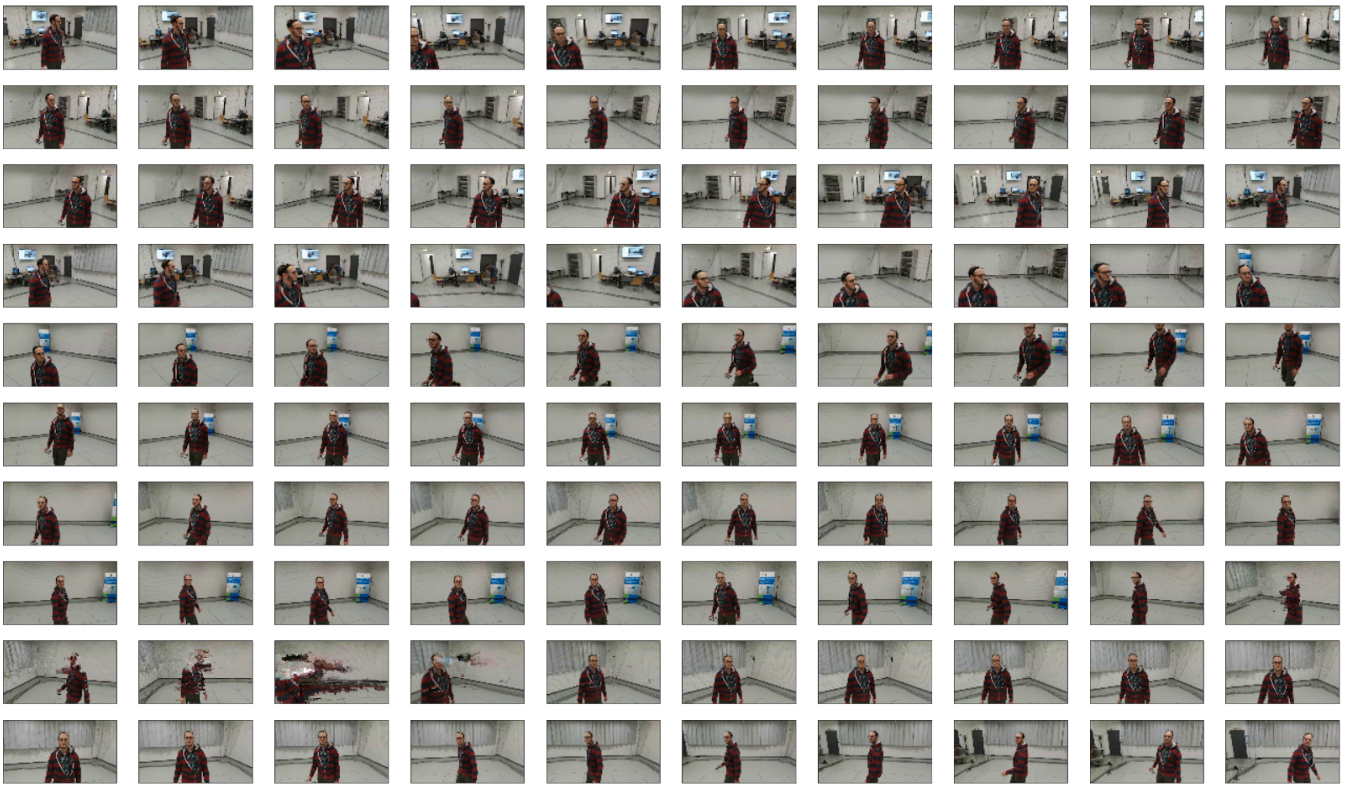
Work in progress:

- Try the models on unknown test sets

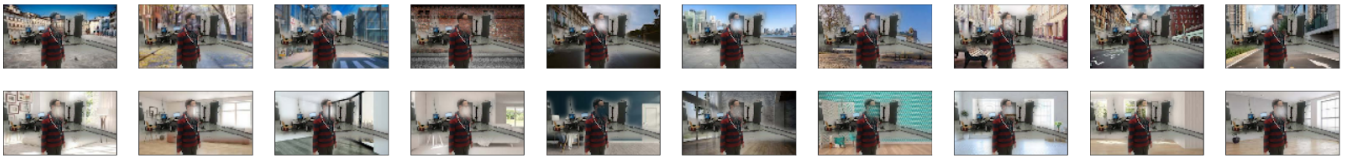
GradCAM application

This chapter analyzes how the last two models presented behave with GradCAM. Both models are stopped at 30 epochs, in order to have a more similar loss between the two.
A subset of the dataset that will be used for the next experiments is shown below.

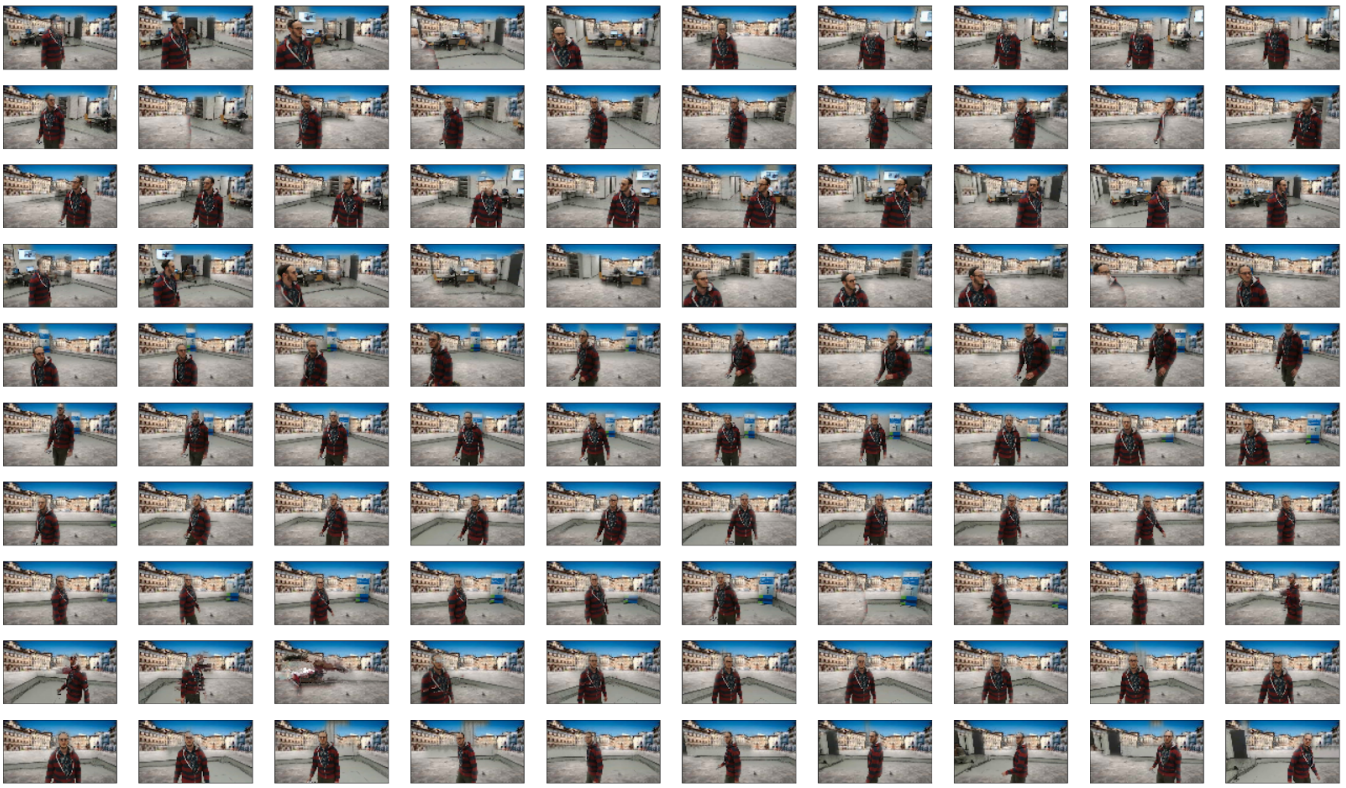
Original background, 100 images (skip 12)



20 backgrounds, 1st image



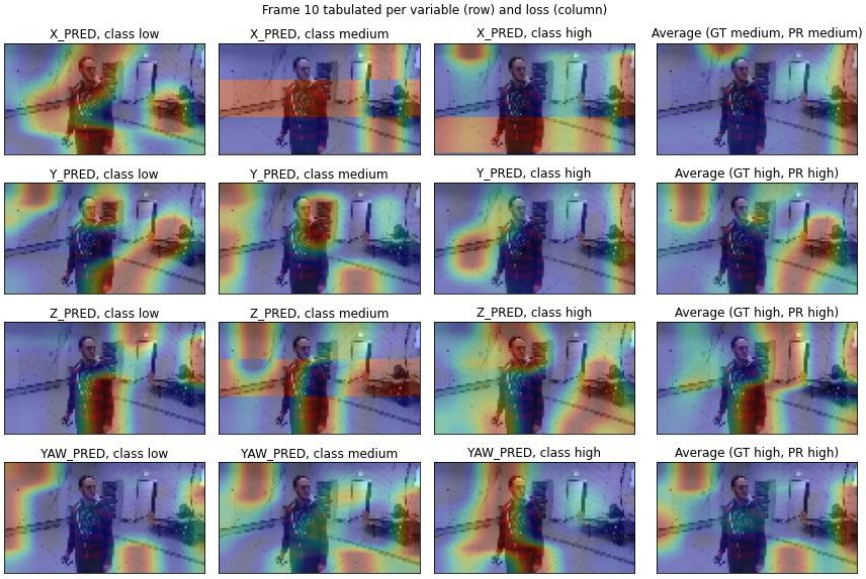
1st background, 100 images



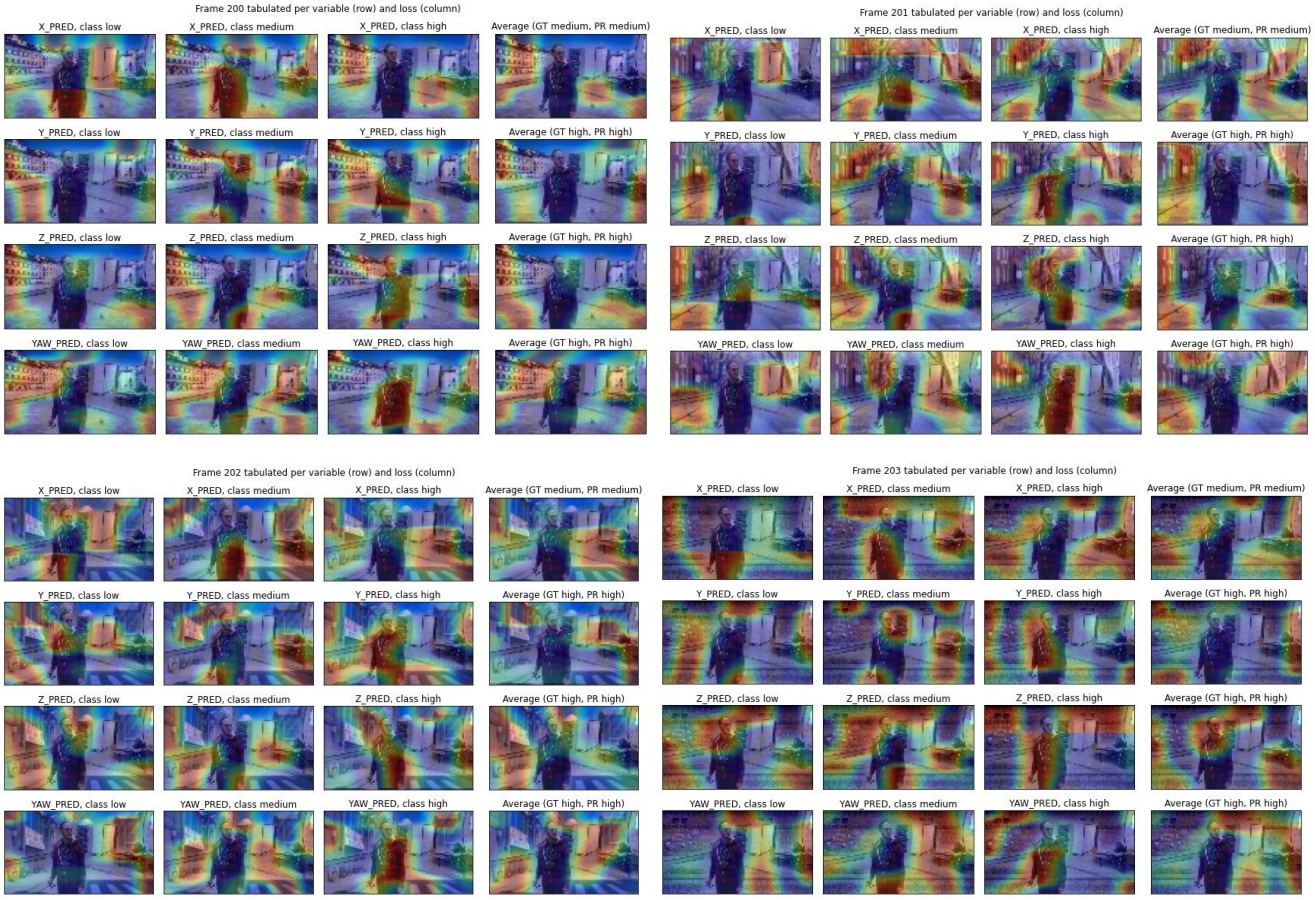
GradCAM application on both models is shown in the following examples. Each image is divided into a grid with the **variables on the rows** (x, y, z, w) and the **classes on the columns** (low, medium, high, average) such that each pair is formed. Above the average class, also ground truth and prediction for the respective input and variable is shown. Analyzing the charts, the reader should mainly consider the **GradCAM heatmap that corresponds to the predicted class** (and the ground truth, if it differs), since the others do not make very much sense. We assume - but it might be incorrect - that the more the person is under the heatmap, the better the model has detected it.

Example 1

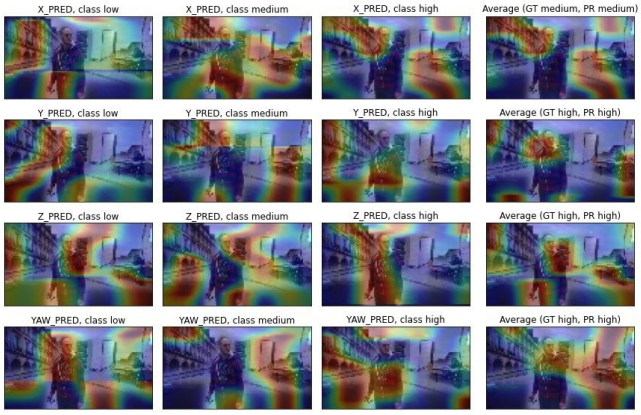
Arena background model



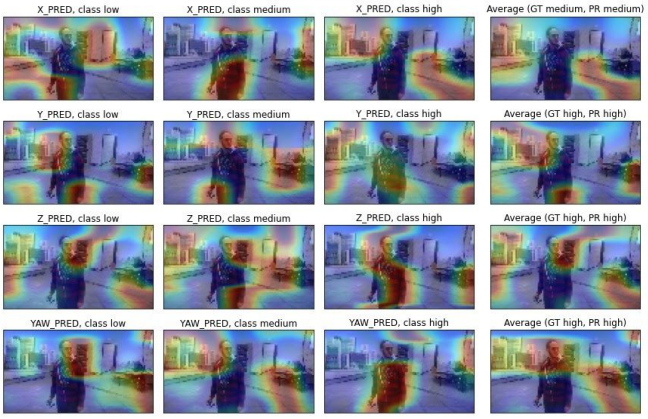
20 backgrounds model



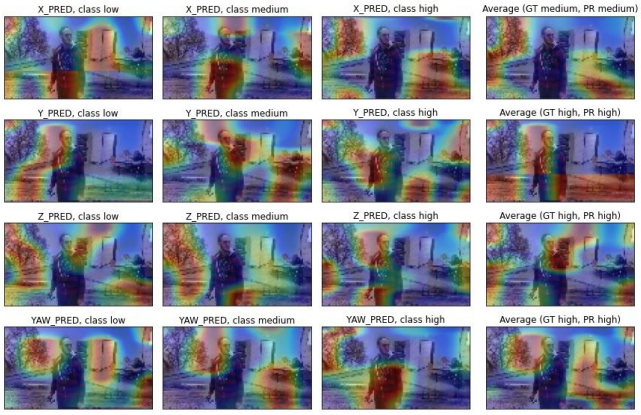
Frame 204 tabulated per variable (row) and loss (column)



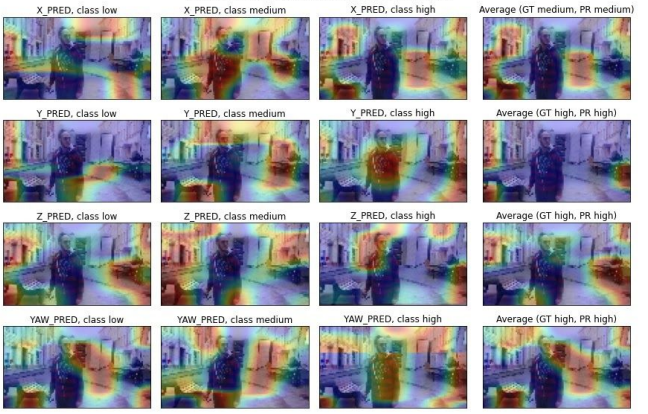
Frame 205 tabulated per variable (row) and loss (column)



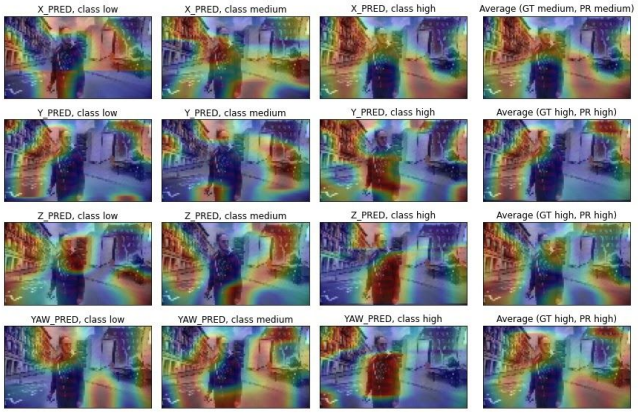
Frame 206 tabulated per variable (row) and loss (column)



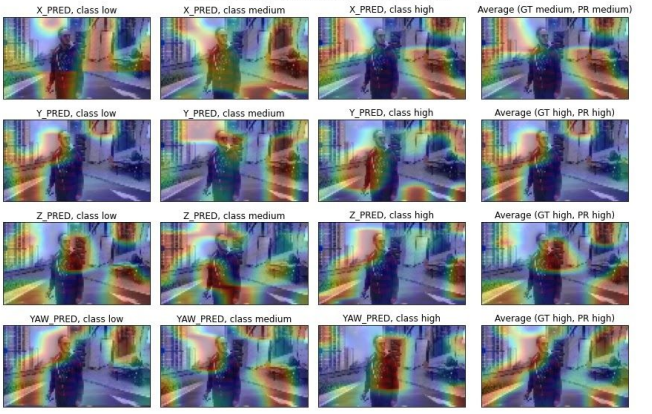
Frame 207 tabulated per variable (row) and loss (column)



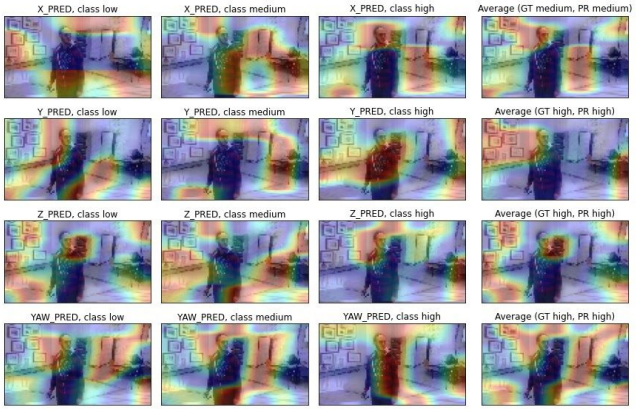
Frame 208 tabulated per variable (row) and loss (column)



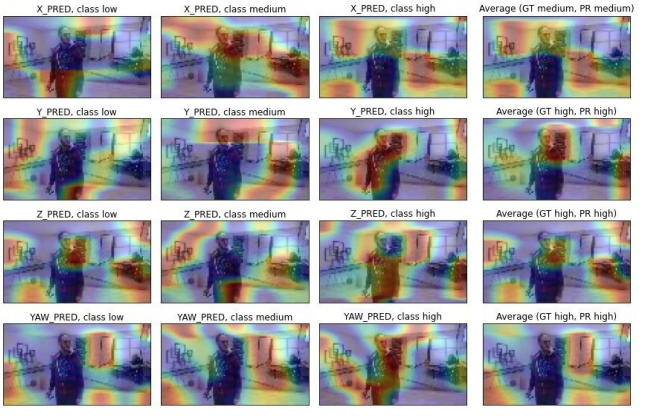
Frame 209 tabulated per variable (row) and loss (column)



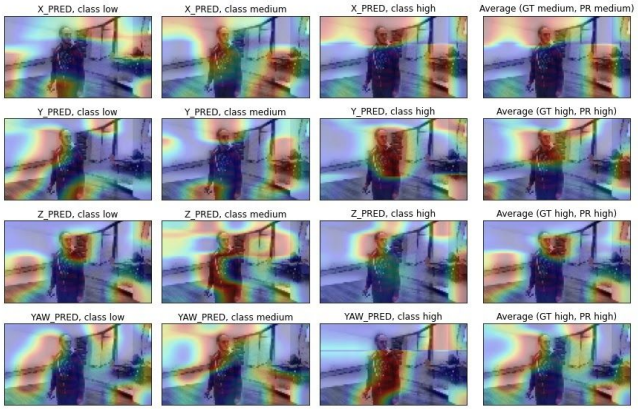
Frame 210 tabulated per variable (row) and loss (column)



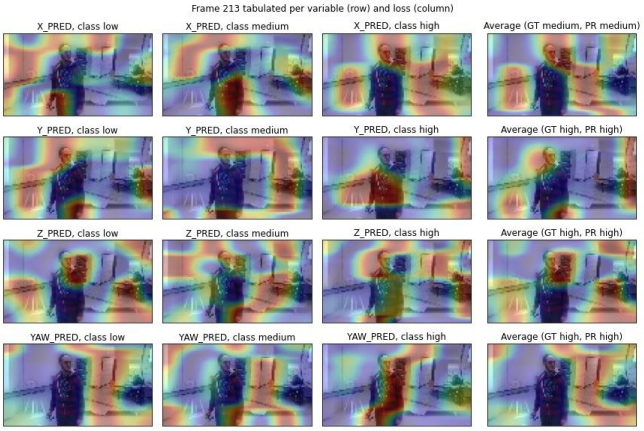
Frame 211 tabulated per variable (row) and loss (column)



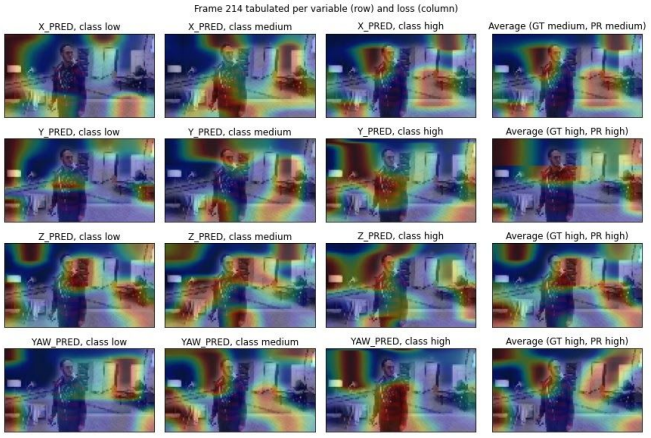
Frame 212 tabulated per variable (row) and loss (column)



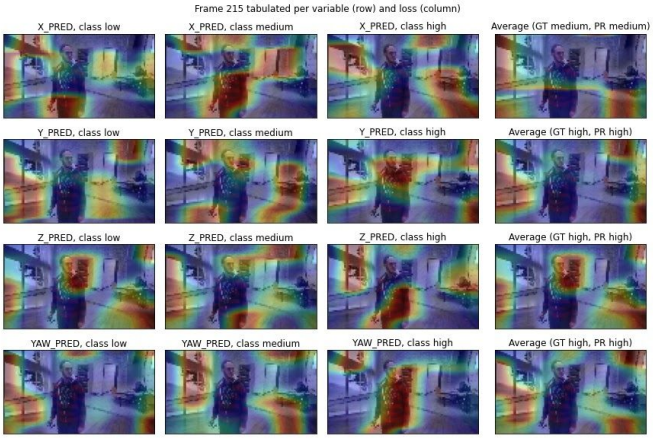
Frame 213 tabulated per variable (row) and loss (column)



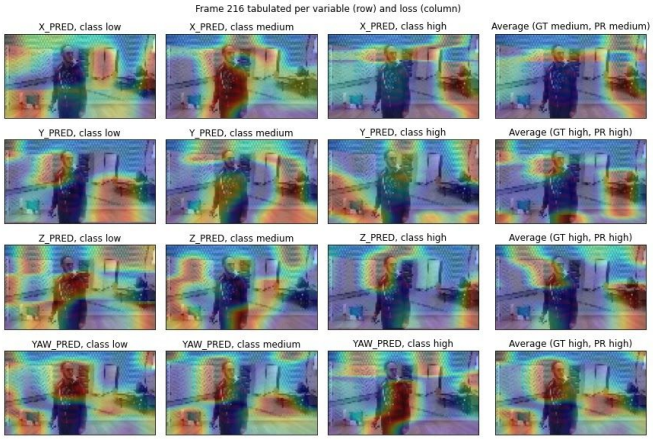
Frame 214 tabulated per variable (row) and loss (column)



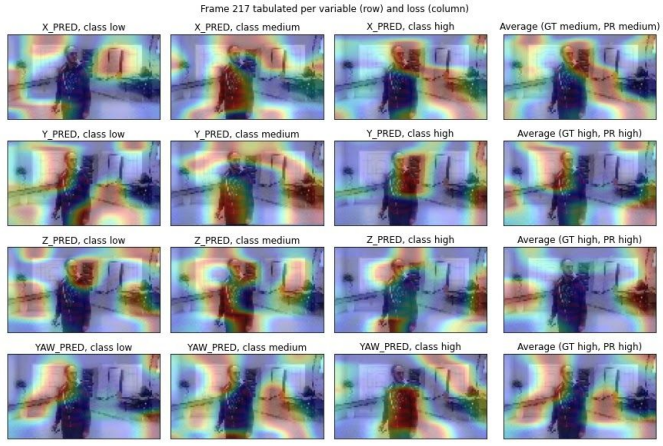
Frame 215 tabulated per variable (row) and loss (column)



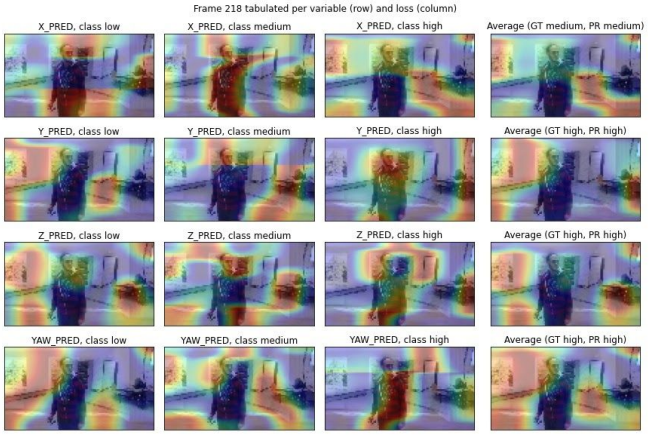
Frame 216 tabulated per variable (row) and loss (column)



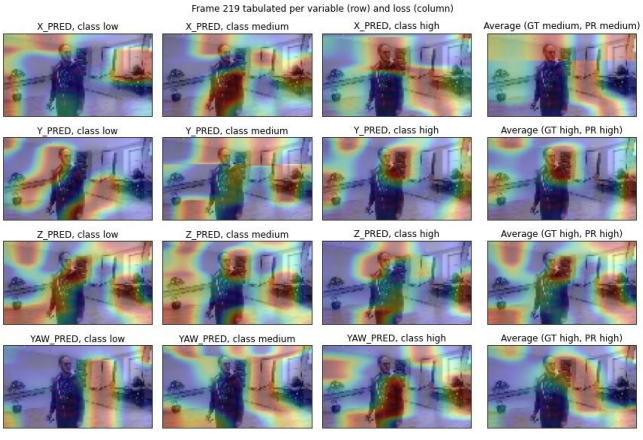
Frame 217 tabulated per variable (row) and loss (column)



Frame 218 tabulated per variable (row) and loss (column)

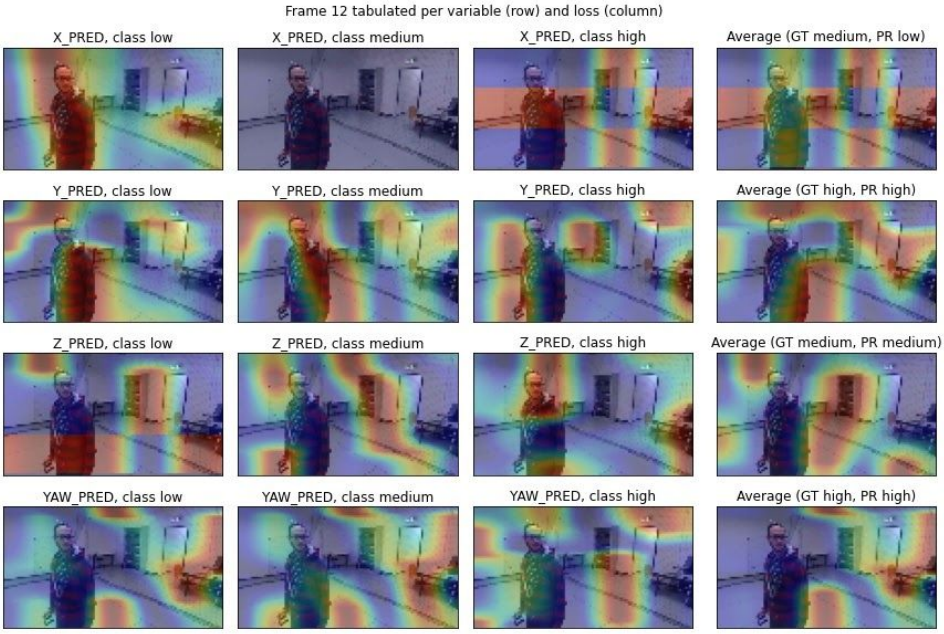


Frame 219 tabulated per variable (row) and loss (column)

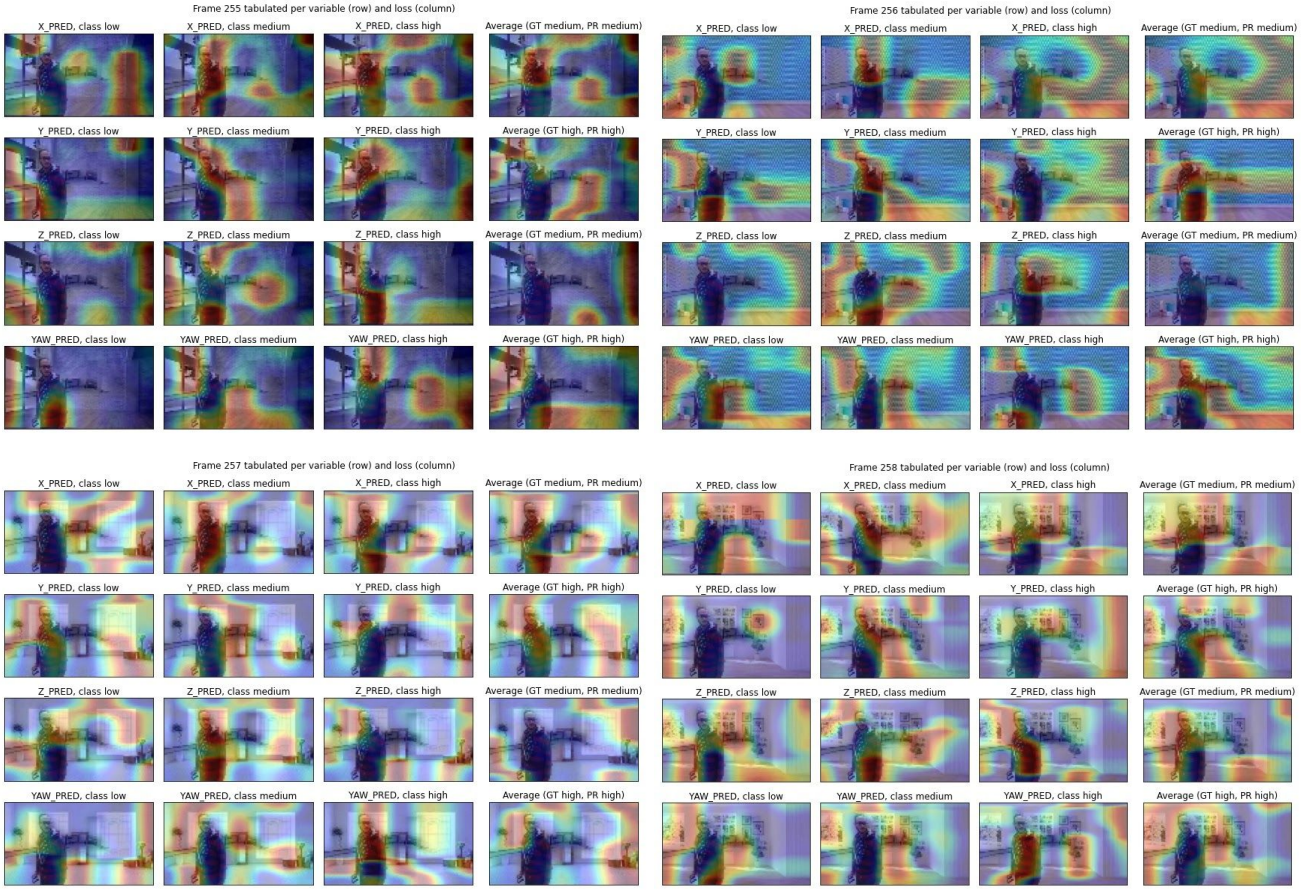


Example 2

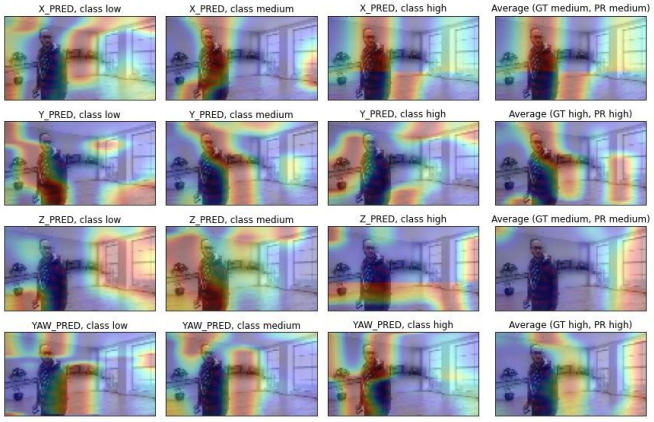
Arena background model



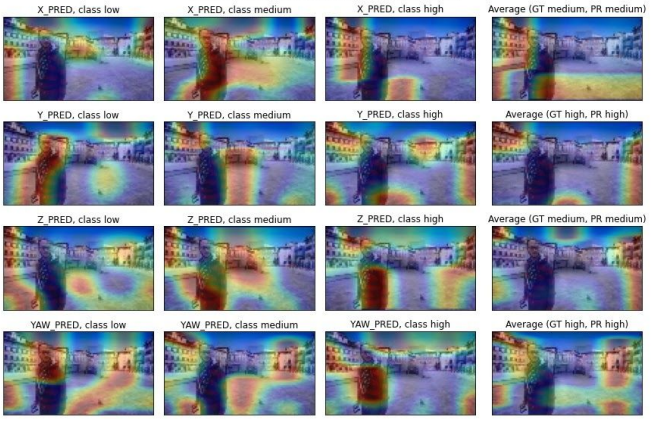
20 backgrounds model



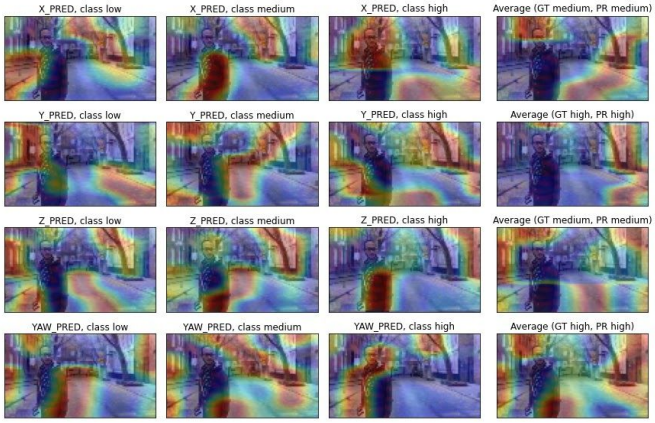
Frame 259 tabulated per variable (row) and loss (column)



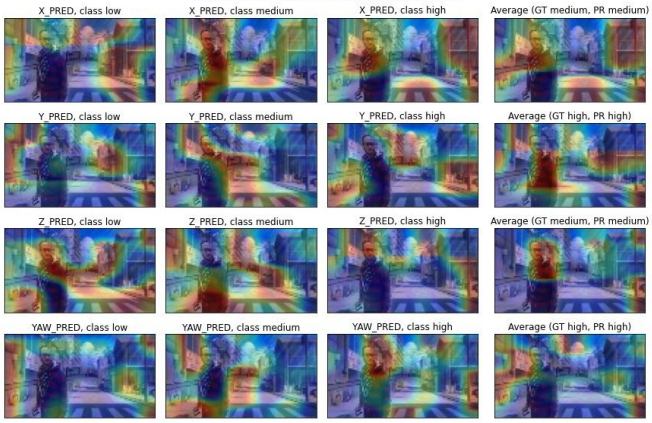
Frame 240 tabulated per variable (row) and loss (column)



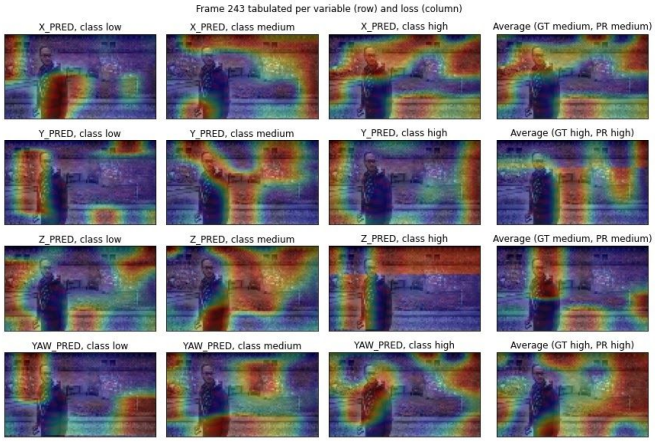
Frame 241 tabulated per variable (row) and loss (column)



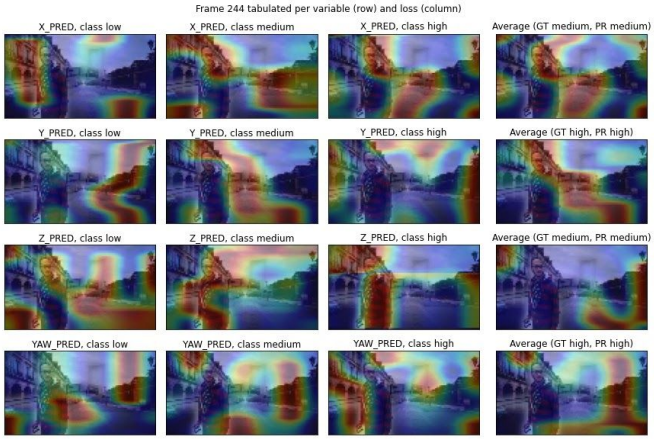
Frame 242 tabulated per variable (row) and loss (column)



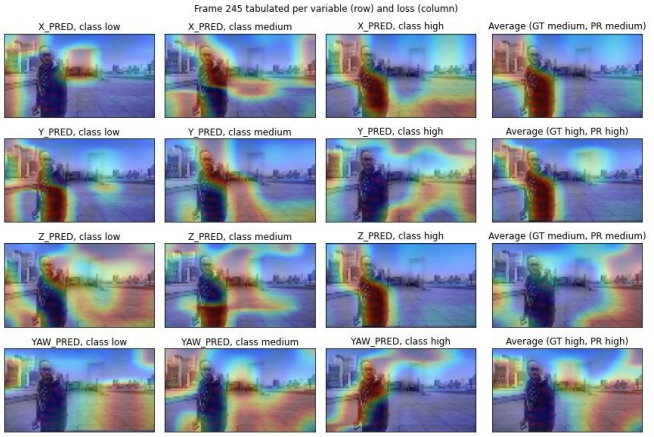
Frame 243 tabulated per variable (row) and loss (column)



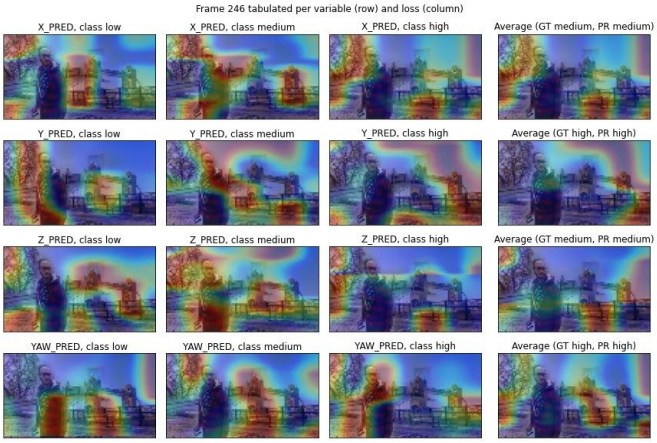
Frame 244 tabulated per variable (row) and loss (column)



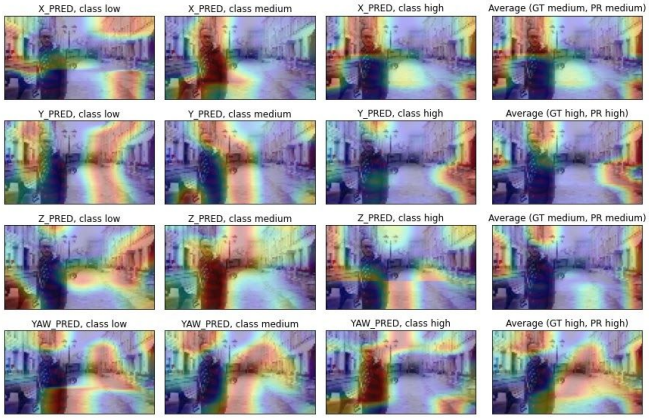
Frame 245 tabulated per variable (row) and loss (column)



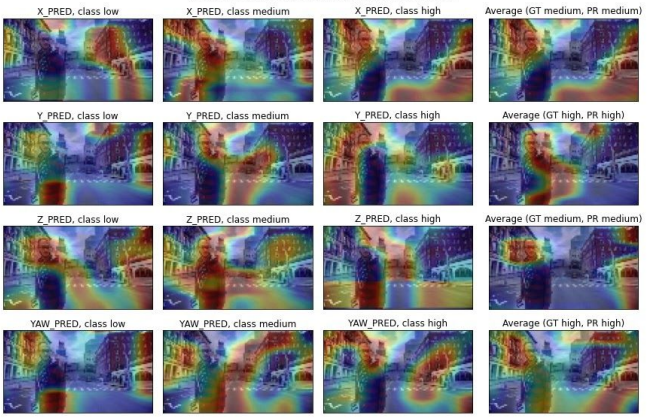
Frame 246 tabulated per variable (row) and loss (column)



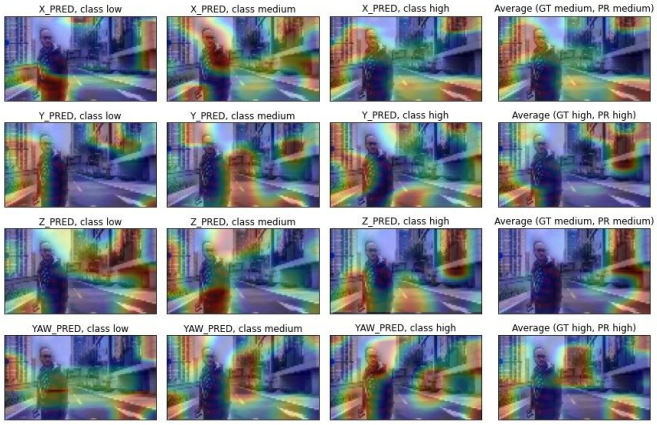
Frame 247 tabulated per variable (row) and loss (column)



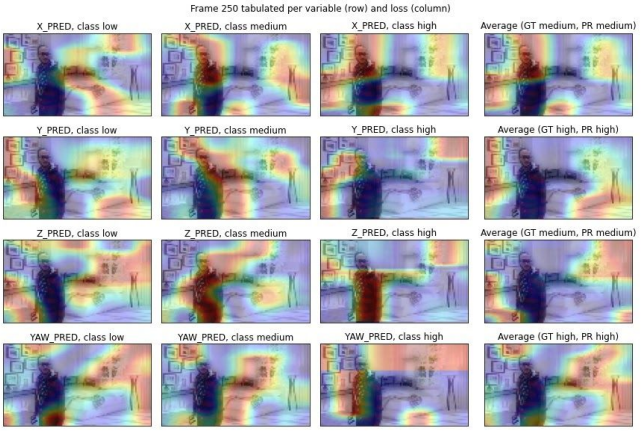
Frame 248 tabulated per variable (row) and loss (column)



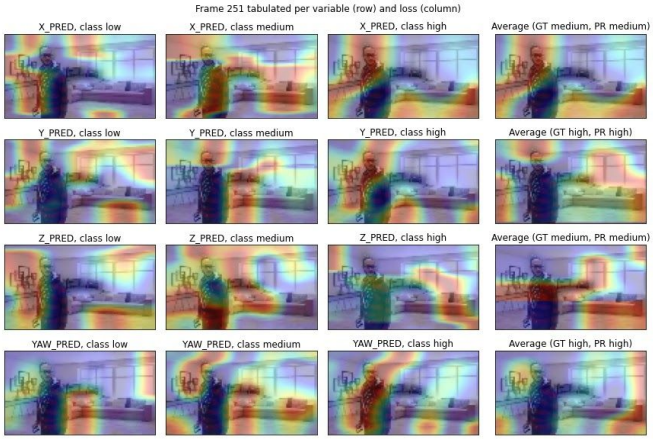
Frame 249 tabulated per variable (row) and loss (column)



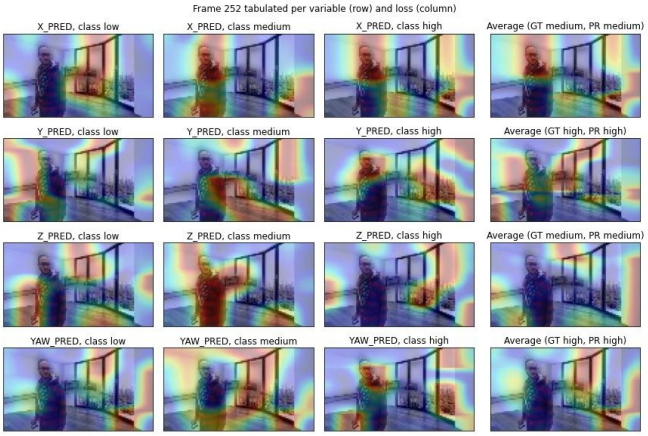
Frame 250 tabulated per variable (row) and loss (column)



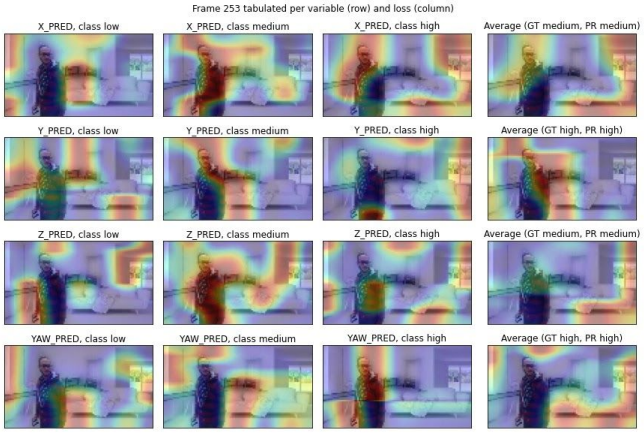
Frame 251 tabulated per variable (row) and loss (column)



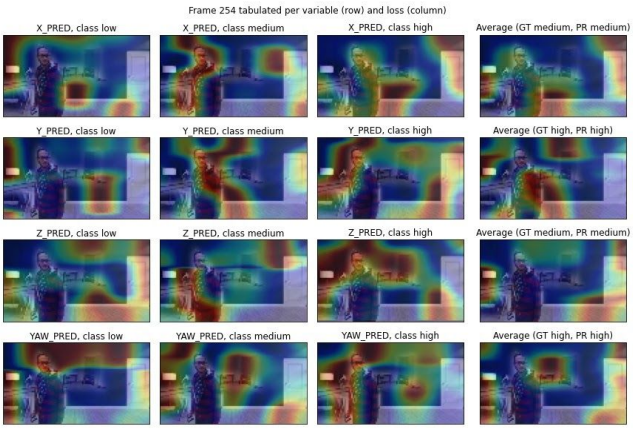
Frame 252 tabulated per variable (row) and loss (column)



Frame 253 tabulated per variable (row) and loss (column)



Frame 254 tabulated per variable (row) and loss (column)



Test set performance comparison

Which follows is the result of running Keras `evaluate()` function on specified models and data sets.

Arena model on the **original test set**
`test_loss: 2.5170`
`x_loss: 0.658, y_loss: 0.446, z_loss: 0.452, w_loss: 0.961`
`x_accur: 0.712, y_accur: 0.833, z_accur: 0.821, w_accur: 0.723`

20backgrounds model on the **original test set**
`test_loss: 3.3068`
`x_loss: 0.908, y_loss: 0.695, z_loss: 0.702, w_loss: 1.002`
`x_accur: 0.722, y_accur: 0.822, z_accur: 0.833, w_accur: 0.711`

20backgrounds model on the **20backgrounds test set**
`test_loss: 3.7442`
`x_loss: 1.084, y_loss: 0.734 z_loss: 0.856, w_loss: 1.069`
`x_accur: 0.619, y_accur: 0.761, z_accur: 0.746, w_accur: 0.630`

Regarding the last evaluation, please note that in the test set it is more difficult to isolate the person from the background since, in most of the frames, it wears a white shirt which is similar to the room colors. So the 20backgrounds test set is very imprecise.
More tests on webcam-provided images will be conducted soon.