MURA TASK

DIMITRIOS FOUNTAS

THIS FILE'S PURPOSE IS TO EXPLAIN THE IMPLEMENTATION OF THE MODELS.

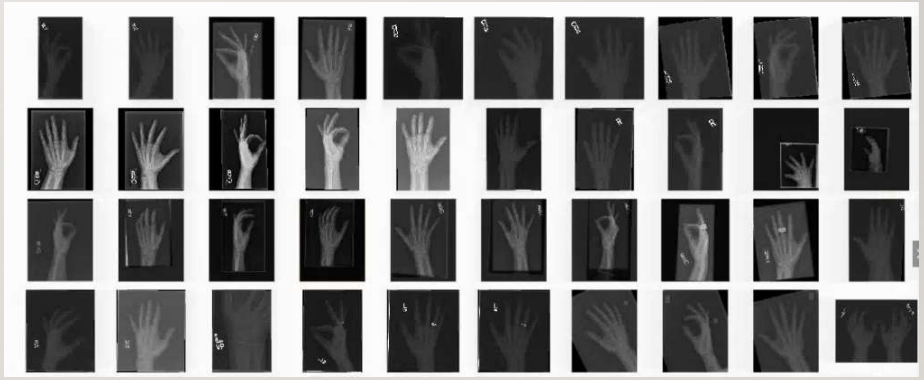
FOR MORE TECHNICAL INFORMATION, PLEASE STUDY THE JUPITER NOTEBOOK PROVIDED AT THE LINK:

HTTPS://COLAB.RESEARCH.GOOGLE.COM/DRIVE/1HHADS9WOASHUB0PRG5_WKXAIUXBSO-PB?USP=SHARING

DATA AUGMENTATION

Choosing the correct Data Augmentation will have critical impact to the performance of the model. We can observe that many images are cropped, others are zoomed out and more importantly some of them are left or right rotated, so we choose to augment the images based on those observations.

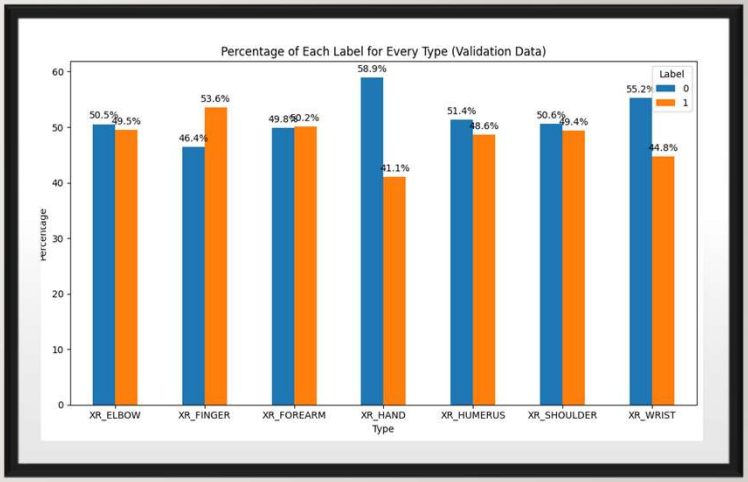
Clearly, zooming in and out is critical because we can observe that there are pictures that the actual information is at one of the corners.

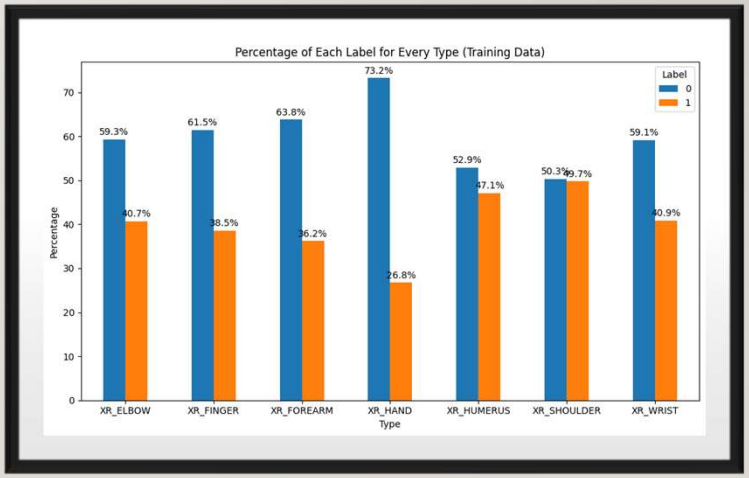


INVESTIGATING TRAIN AND VALIDATION DATA.

WE CAN EASILY OBSERVE THAT WE CAN SEE THAT EXCEPT FOR THE HAND. OUR POSITIVE TO NEGATIVE RATIO IS BALANCED BETWEEN TYPES.

OUR VALIDATION DATA ARE MORE BALANCED WHEN IT COMES TO THE POSITIVE TO NEGATIVE RATIO COMPARED TO OUR TRAINING DATA.





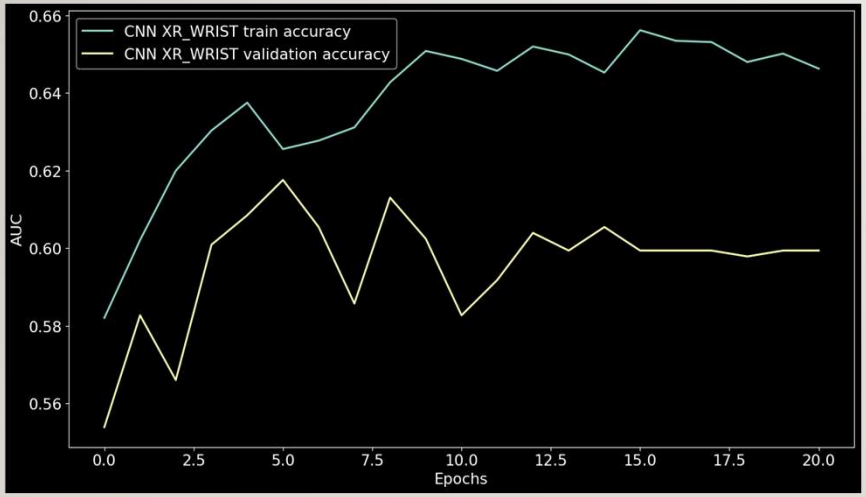
CUSTOM CNN WITHOUT PRE - TRAINED

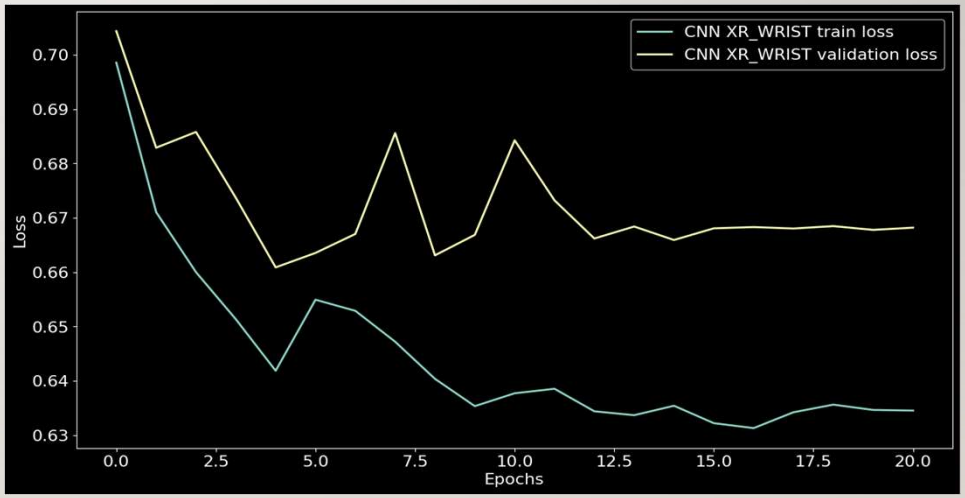
We create an CNN that starts With 4 convolutions, all of them have kernel size (3,3) but 2 of them have strides (2,2) while the other two have dilation (2,2) to get both the benefits of the stride and the dilation.

We do Batch Normalazation before each maxpooling, and we add a dropout layer to help avoid overfitting. Finally, we flatten the output and pass it through some dense layers. Since our problem is binary, we use at the last dense layer the sigmoid activation function and the metric we try to optimize is the binary cross entropy. You can observe that our filters are very small, because we have RBG images and when we eventually flatten them, we will not be able to use a dense MLP with more than 10 units because of memory restrictions.

TRAINING OUR FIRST MODEL ON THE WRIST.

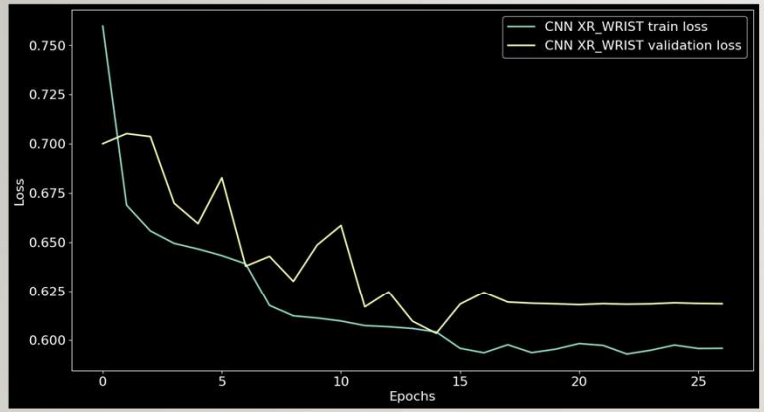
The results of the model are discouraging, our model overfits really fast and is able to do just a little better than guessing randomly.

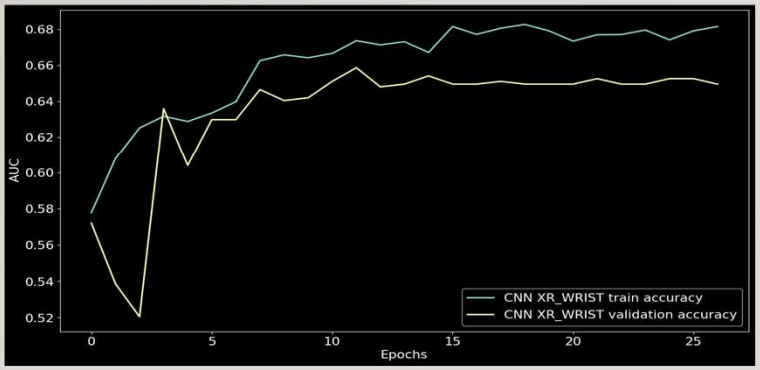





```
Train Loss      : 0.63453
Validation Loss: 0.66818
Train Loss      : 0.62214
---
Train Accuracy      : 0.64631
Validation Accuracy: 0.59939
Test Accuracy       : 0.69672
```

IMPROVING CUSTOM CNN WITHOUT PRE – TRAINED BY USING SELF ATTENTION.





We add a self attention Layer that reduces the memory we need to train the Dense Layers, That allows us to create more filters at the convolutions and capture more information.

By adding a self attention Layer to our previous CNN, we cut the time of the training significantly while at the same time we increase validation accuracy by 5% and test accuracy by 7%.

Our model starts overfitting at the 14th epoch.

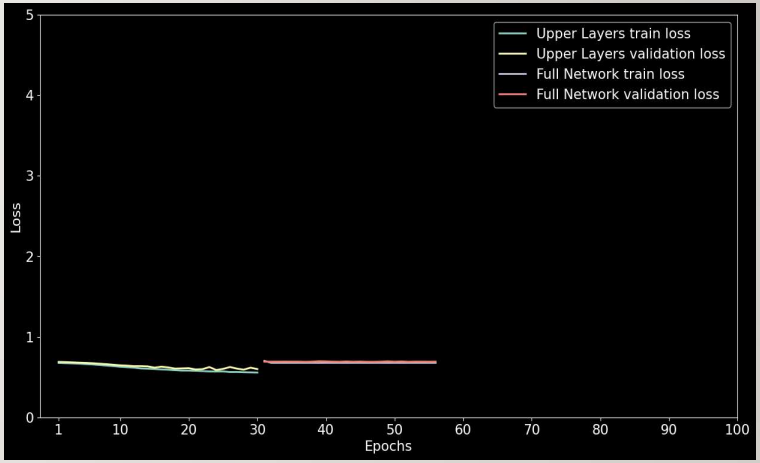
```
Train Loss      : 0.59570
Validation Loss: 0.61832
Train Loss      : 0.55814
---
Train Accuracy      : 0.68140
Validation Accuracy: 0.64947
Test Accuracy       : 0.71926
```

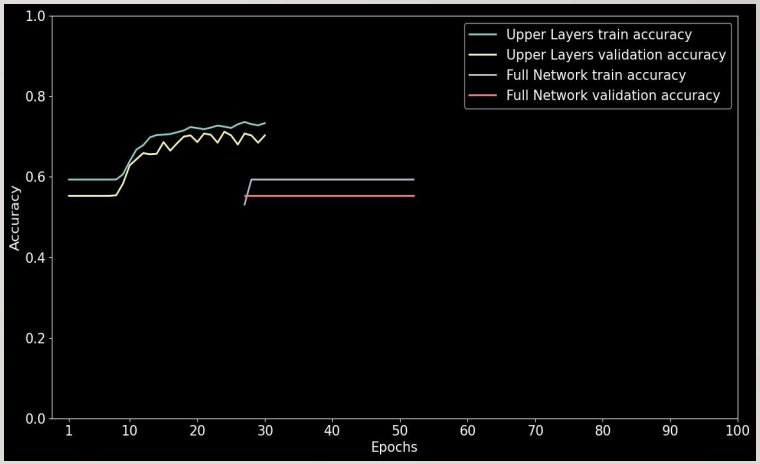
CUSTOM CNN WITH PRE - TRAINED

We use the pre trained model InceptionV3 and we tune it to the task by removing its out-pout layer, doing a max pooling and stacking three dense layers. First, we train the lower Layers for 30 epoch and then we try to train 70 epoch the full model.

```
Upper Layers Evaluation:
---
Train Loss      : 0.55585
Validation Loss: 0.59859
---
Train Accuracy      : 0.73302
Validation Accuracy: 0.70258
Full Network Evaluation:
---
Train Loss      : 0.67609
Validation Loss: 0.69147
---
Train Accuracy      : 0.59287
Validation Accuracy: 0.55235

---
Test Loss    : 0.68161
Test Accuracy: 0.57582
```

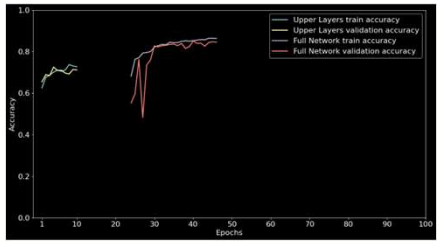


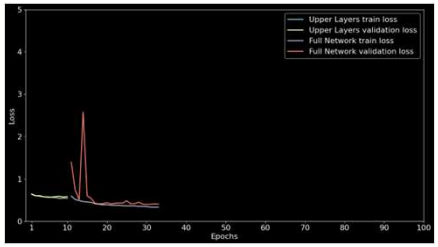


CUSTOM CNN WITHOUT PRE – TRAINED IMPROVEMENT.

The previous results were discouraging, the full model had a worst performance than our simple CNN models, and the lower Layers model that trained only the low layers while keeping the pre-trained weights, improved the accuracy only by 5%.

We now add an Add gate that connects the out-pouts of the first MLP of the lower Layers to the Last one and we retrain the same model.






```
Upper Layers Evaluation:
---
Train Loss      : 0.54351
Validation Loss: 0.57970
---
Train Accuracy      : 0.72675
Validation Accuracy: 0.71168
Full Network Evaluation:
---
Train Loss      : 0.33246
Validation Loss: 0.40380
---
Train Accuracy      : 0.86326
Validation Accuracy: 0.84522

---
Test Loss     : 0.37534
Test Accuracy: 0.84939
```

CUSTOM CNN WITH PRE – TRAINED AND ADD GATE

While the training of the Upper Layers did not result in a better accuracy that the upper Layers of the previous model. The full model ended up improving the accuracy of our full model by almost 14%.