Pretraining analysis

The purpose of the pretraining is to stimulate the networks using a smaller dataset in order to identify milestones and the best hyper-parameters to use for this project.

It is also a good practice to at least check if the project is feasible and in what conditions.

For this experiment I will be using a dataset of 15 images for training and 5 for validation per class (62 classes in total). The quantity is determined by the capacity of the GPU of my computer.

Once we have demonstrated that the project is feasible then it will be again executed using a framework with more capabilities in particular in GPU RAM memory that can be provided by Amazon Web Services called SageMaker or another provider. The complete dataset is near to two hundred times larger than we need this kind of support.

Training and definition of epochs

In the paper [?] it is denoted that the dataset is cropped and multiplied by eight by flipping the images horizontally and vertically and rotating the image 90°, 180° and 270° and they train the model with one epoch of this expanded dataset. In my case I will consider the flipping and rotating but I will generate minibatches of 10 images randomly selected and letting the generator randomly perform the flipping and rotation and every minibatch is used four times to train every epoch.

According to the paper 4 architectures were used.

The architectures were created and adapted to the project.

The optimizer used for the training was Adam. I also tried to train using decent gradient, but the results were not good.

The models were initially loaded with imagenet weight as suggested by the paper.

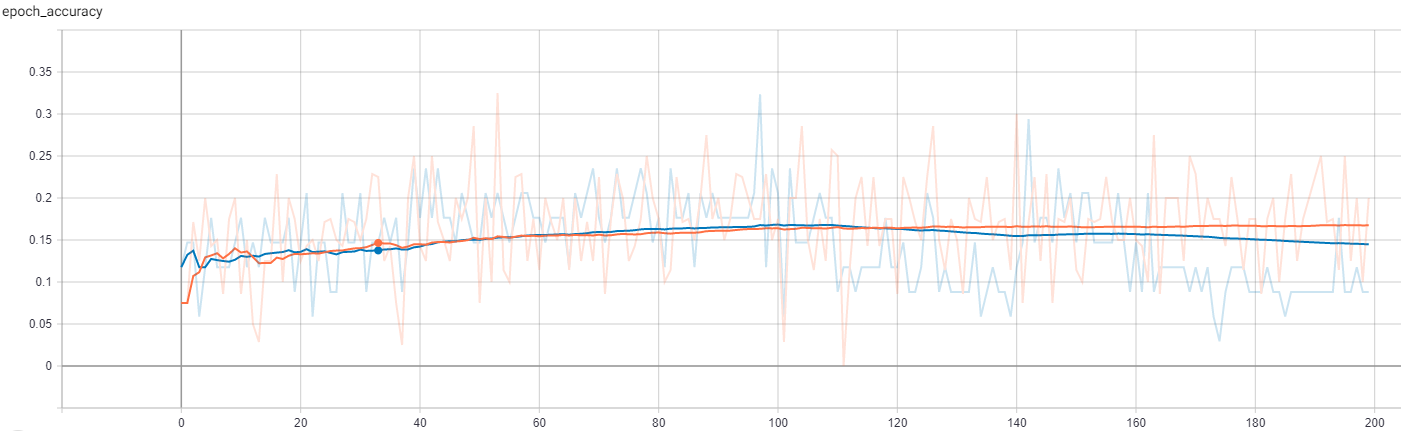
Then, there is just one hyper-parameters to find, it is the learning rate. At this point we are going to avoid other hyperparameters as dropout or modify the learning rate during the training.

The Learning Rate optimization was performed using Tensorboard and doing comparisons between the results.

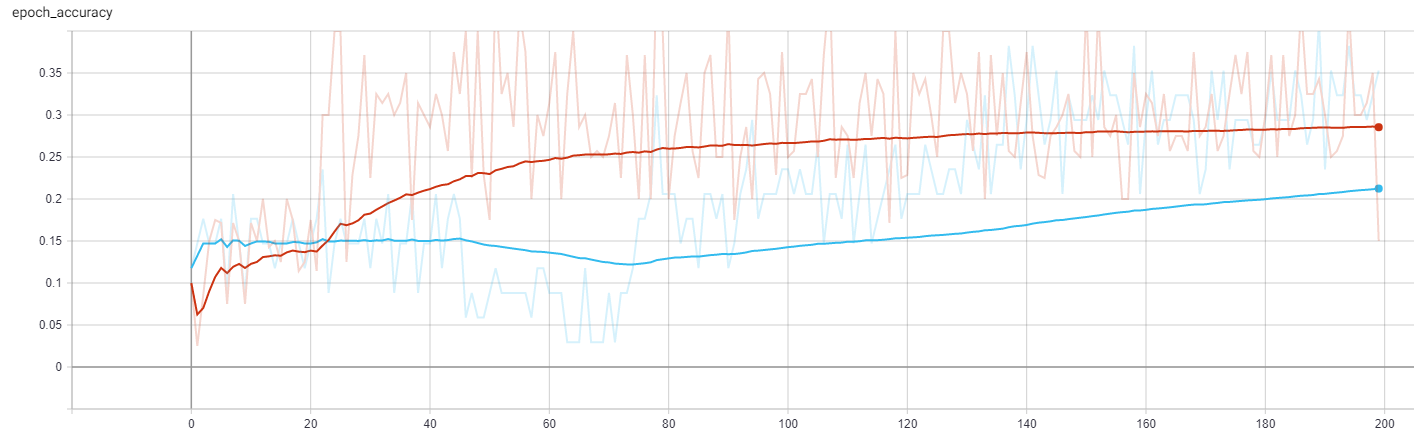
First a code was developed to train the models two hundred epochs starting from a learning rate of 0.5 and divide such learning rate by ten every twenty epochs to analyze the learning rate window from 0.5 to 5x10E-10. The result will be analyzed to check the window were the model shows learning to then focus the analysis in such area.

For DenseNet it was required to modify the learning rate every forty epochs because any visible change appeared using just 20 epochs.

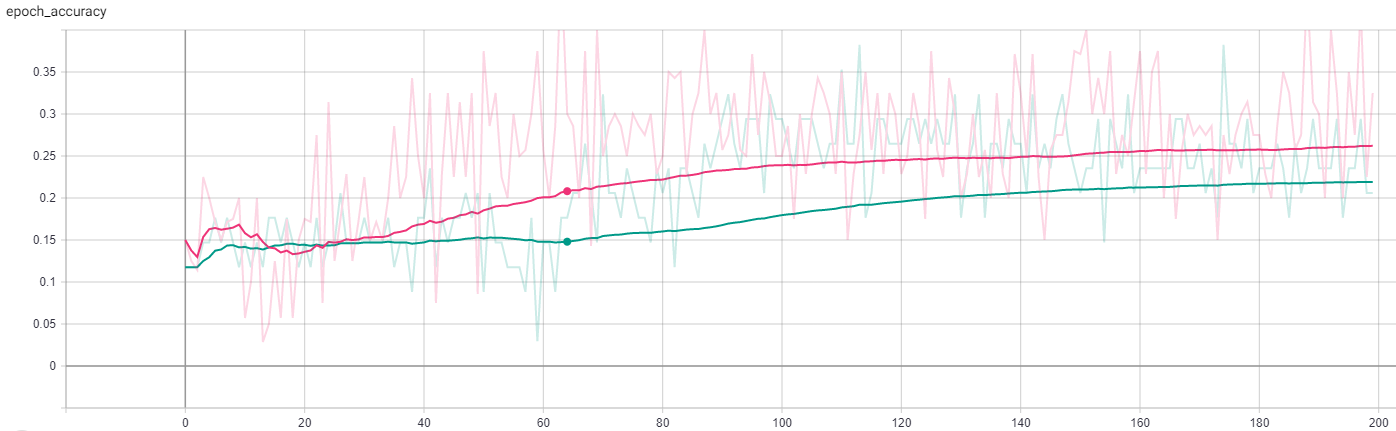
The way to analyze the graphs is making the as smooth as possible and to check were the increasing clear and not too noisy.



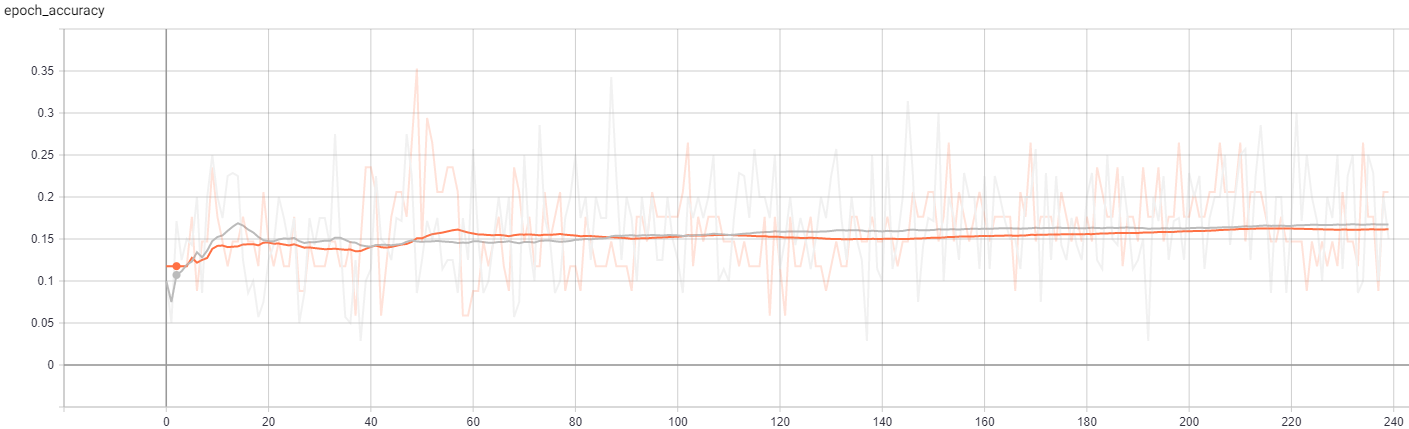
In this example that is the analysis for Resnet152, we can see that the window of improvement goes from the epoch 40 to the epoch 120 that represents the learning rate 0.005 to 0.000005.



For InceptionV3 goes from 0.05 to 0.000005.



For Xception goes from 0.005 to 0.000005.



For DenseNet161 goes from 0.005 to 0.00005.

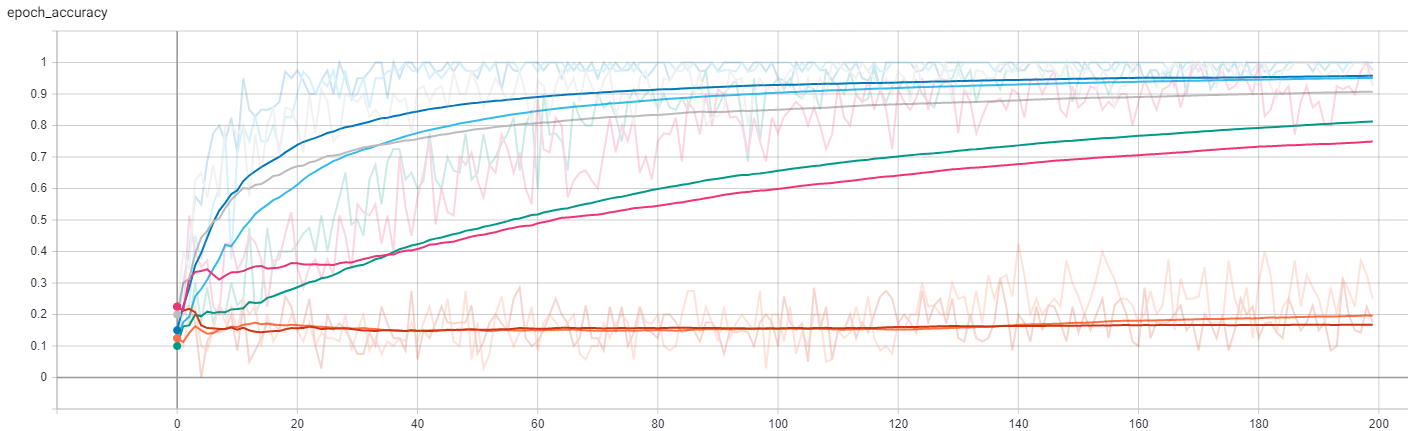
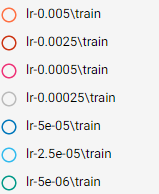
For this experiment I am considering to be able to execute near to 240 epochs in order to be able to have a wide window of analysis for the learning rate. In addition, we are considering for this experiment just to use seven classes to make the training shorter. After having the correct learning rate, the training will be executed using all the sixty two classes and using a more powerful framework to confirm if such learning rate keeps working using all the classes if it isn’t then the experiment has to be performed again from the beginning but using all the classes.

|  |  |
| --- | --- |
| Model | Range to analyze |
| Resnet152 | 0.005 to 0.000005 |
| InceptionV3 | 0.05 to 0.000005 |
| Xception | 0.005 to 0.000005 |
| DenseNet161 | 0.005 to 0.00005 |

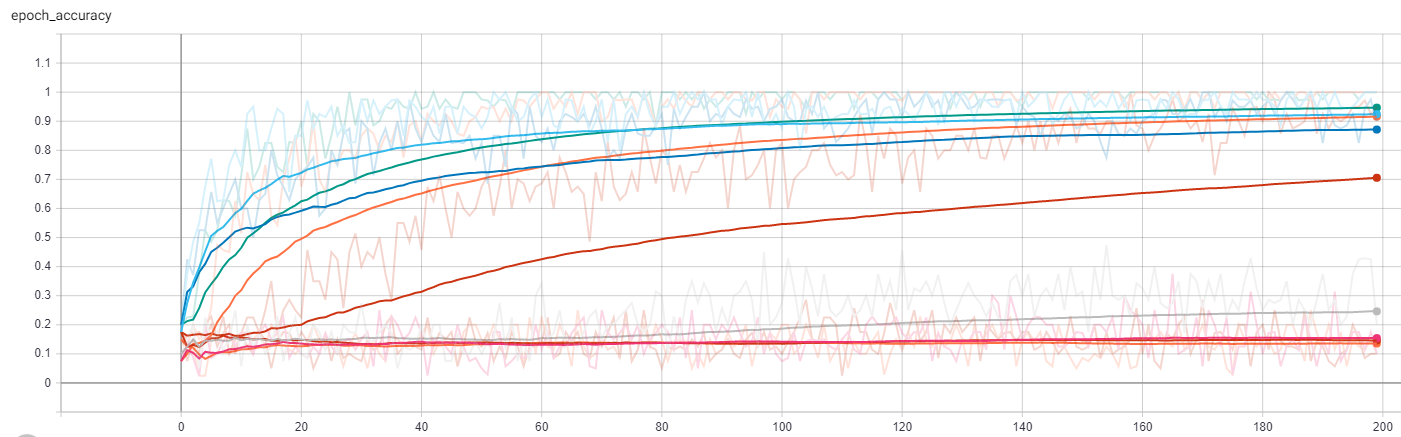
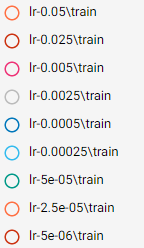
The results are the following

These graphs show just the training accuracy because the assumption is that after using all the dataset the validation value will be close to the training value.

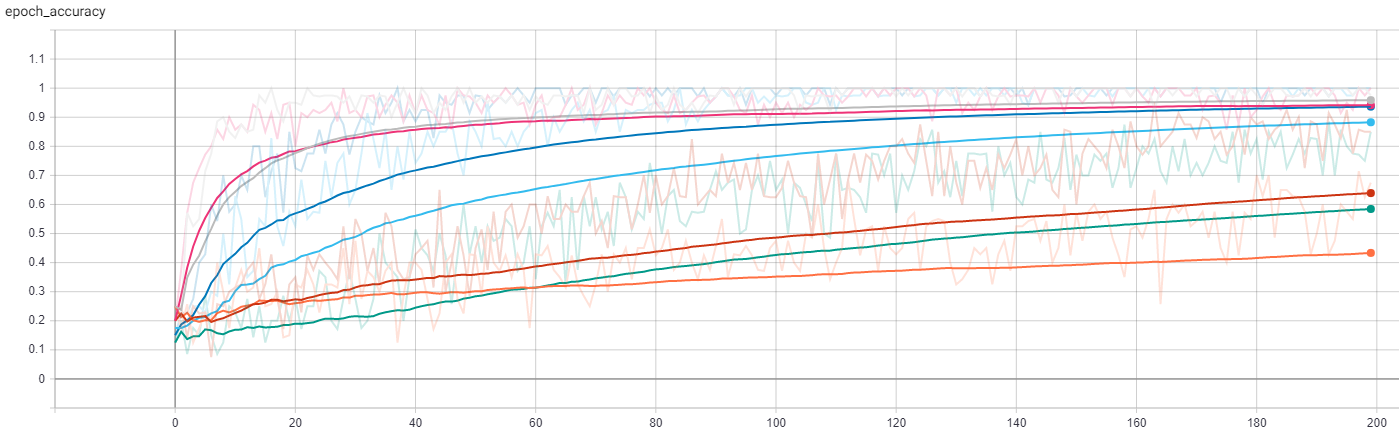
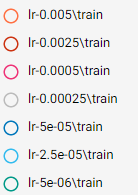
Resnet152



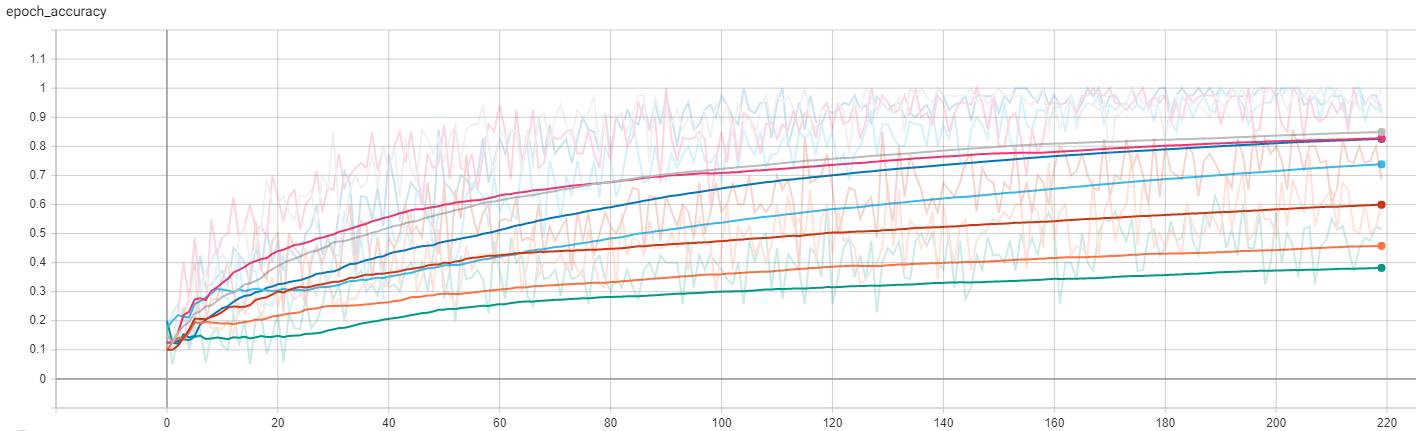
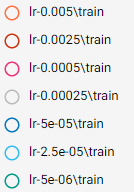
InceptionV3



Xception

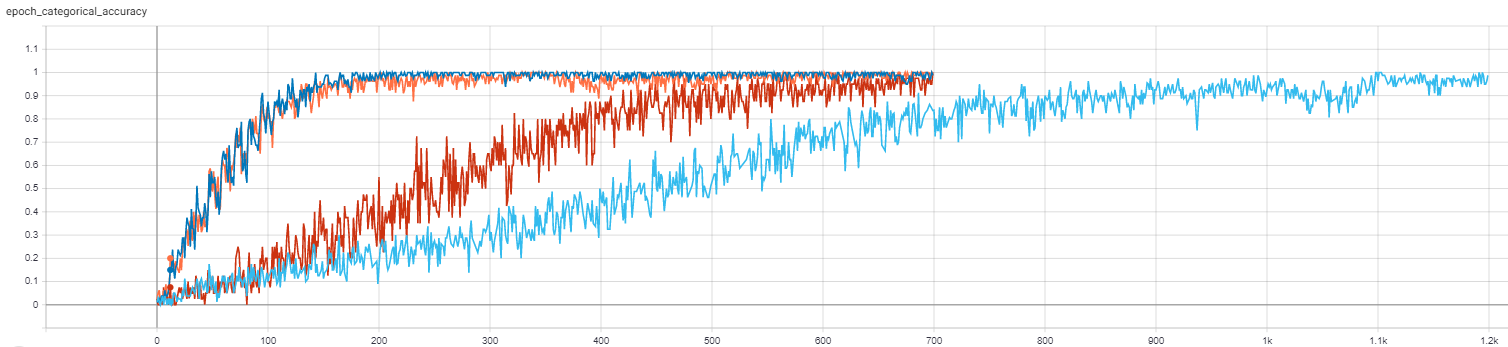


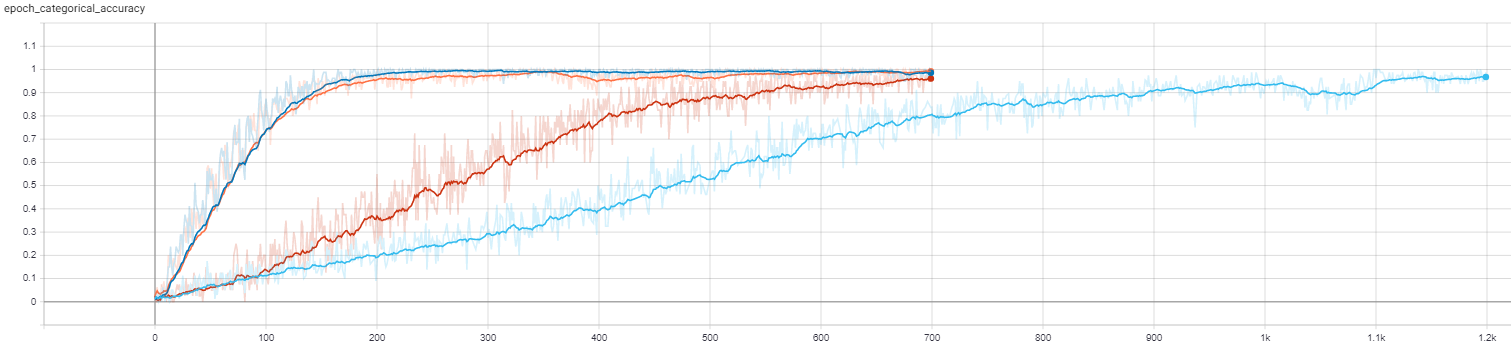
Densenet161



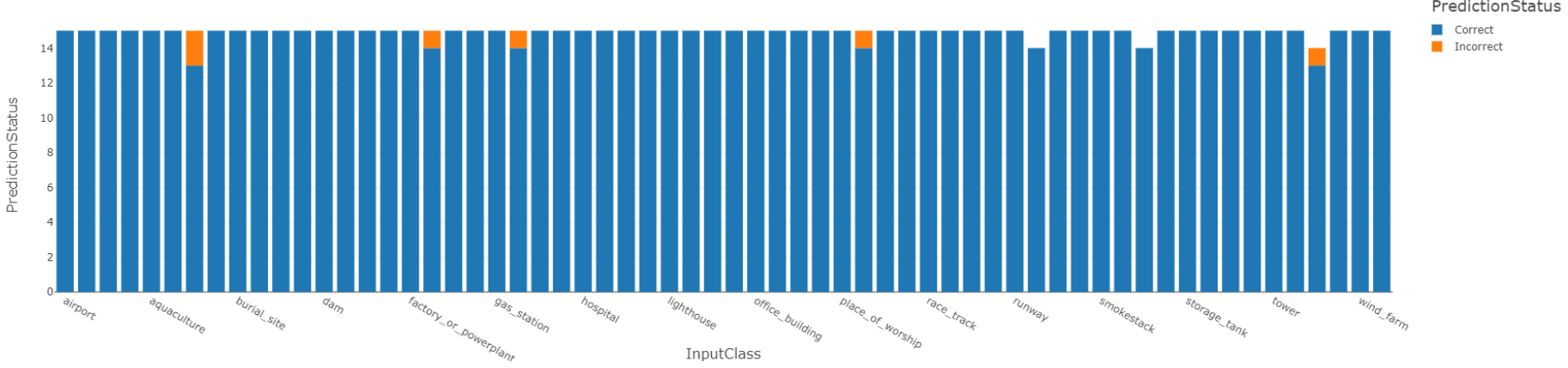
|  |  |
| --- | --- |
| Model | Best LR |
| Resnet152 | 0.00005 |
| InceptionV3 | 0.00005 |
| Xception | 0.00025 |
| DenseNet161 | 0.00025 |

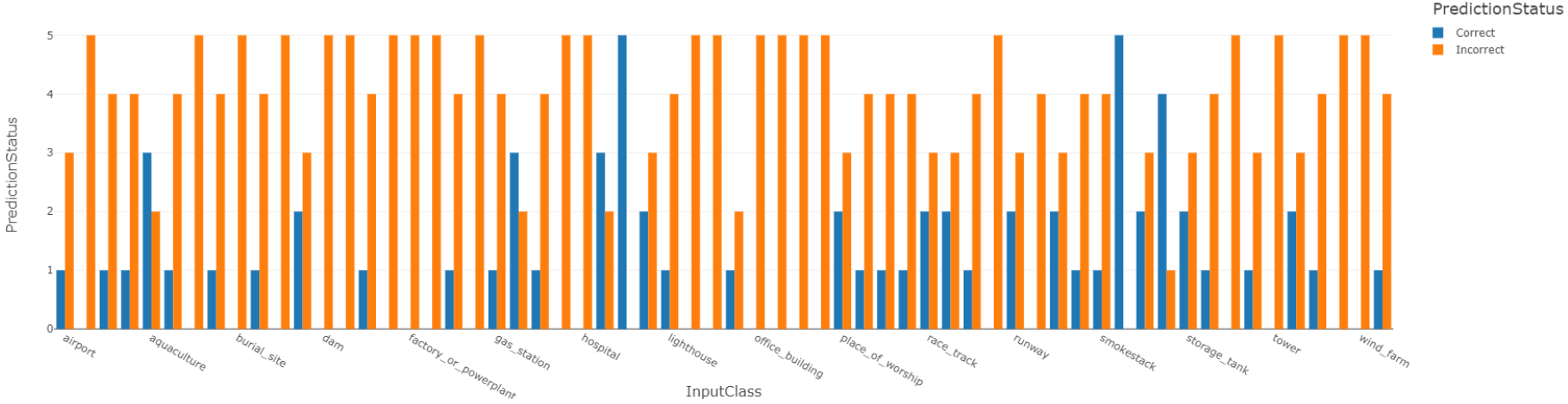
Training Accuracy



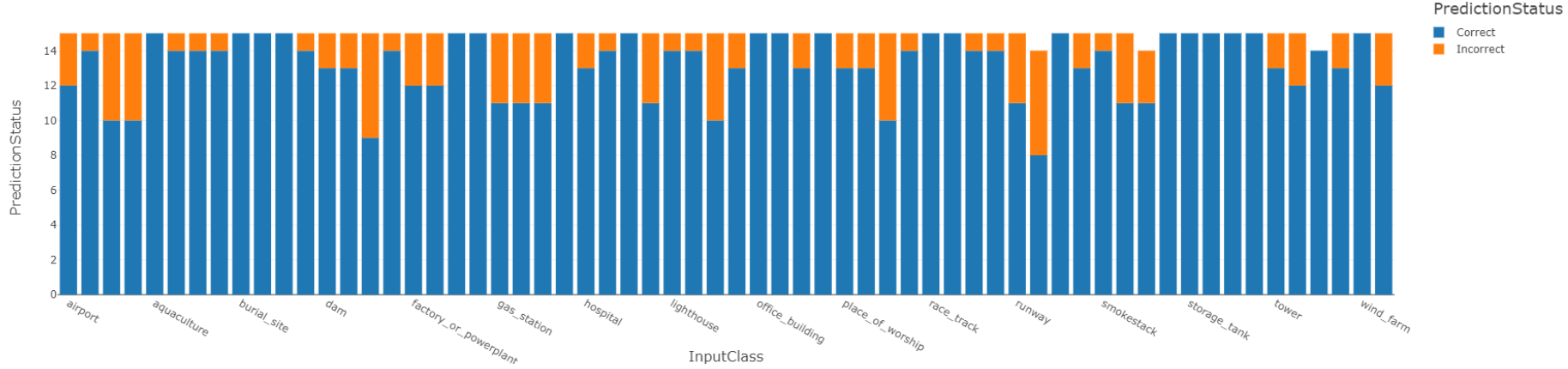


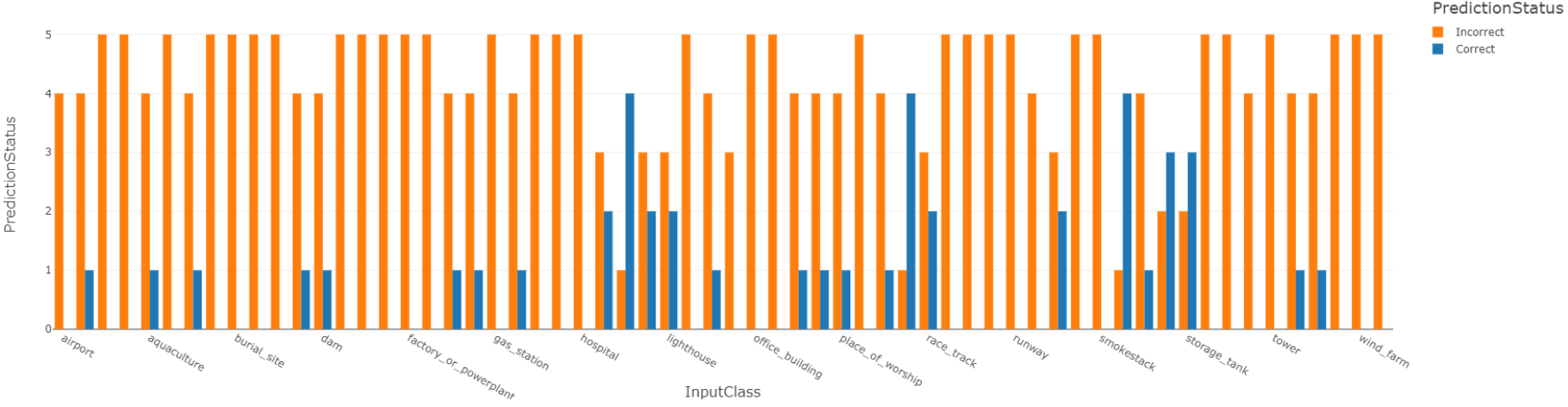
Resnet152



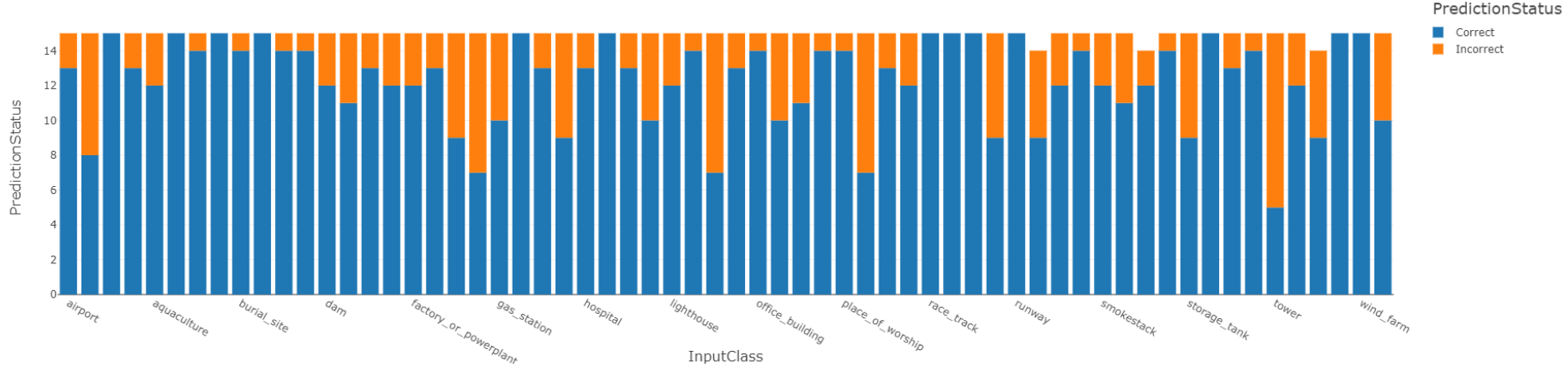


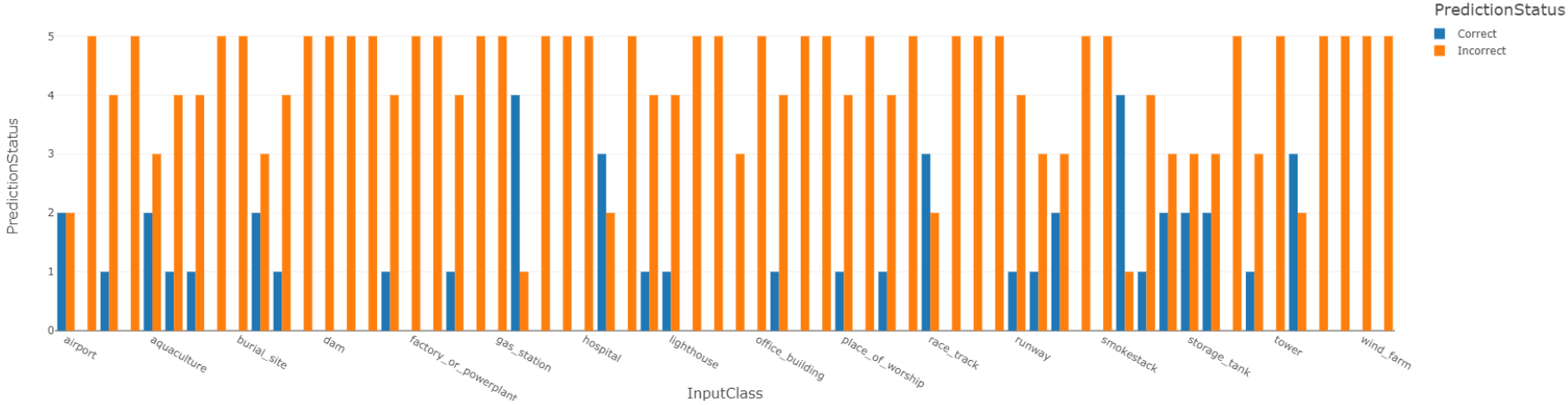
InceptionV3



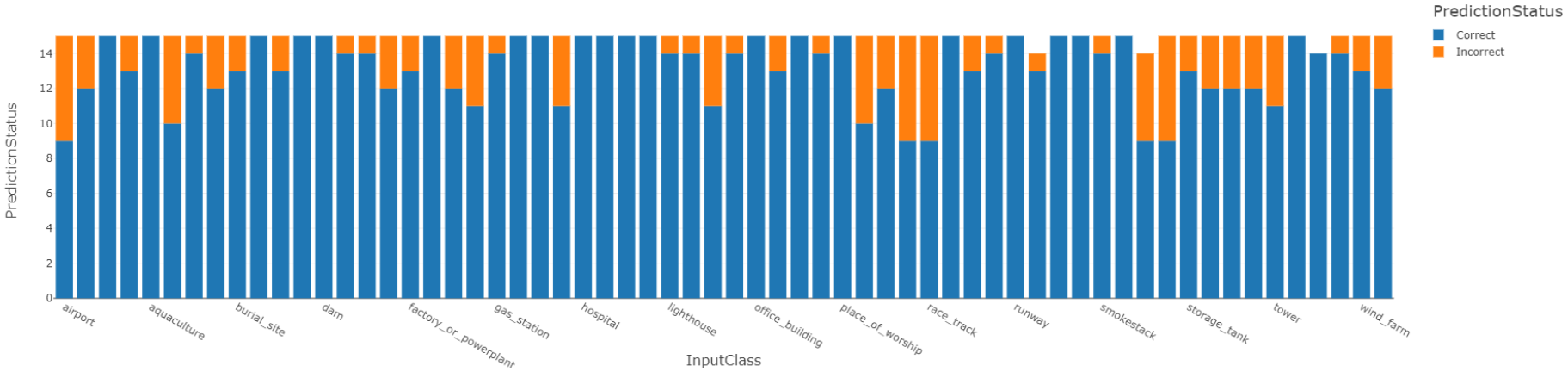


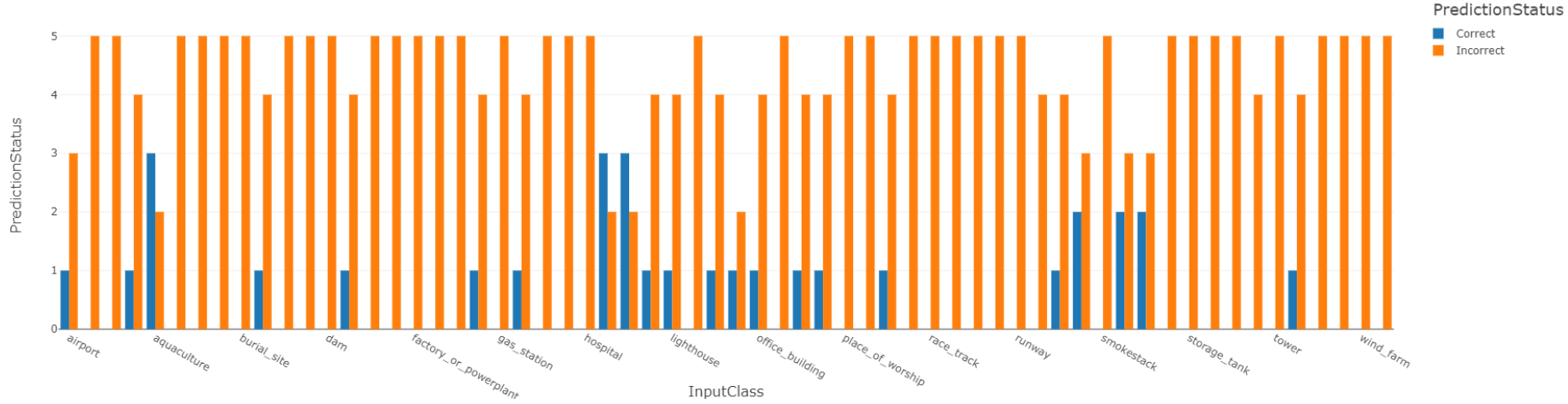
Xception





DenseNet





|  |  |  |
| --- | --- | --- |
| Model | Average\_Accuracy (Training)(higher is better) | Average\_accuracy  (validation) (higher is better) |
| Resnet152 | **0.9934715821812596** | **0.2134408602150538** |
| Inceptionv3 | 0.8842549923195083 | 0.14193548387096772 |
| exception | 0.8195084485407065 | 0.15161290322580648 |
| Densenet161 | 0.8812596006144391 | 0.10295698924731186 |

|  |  |  |
| --- | --- | --- |
| Model | Max\_Accuracy (Training)(higher is better) | max\_accuracy  (validation) (higher is better) |
|  |  |  |
| Resnet152 | **1** | **1** |
| Inceptionv3 | **1** | 0.8 |
| exception | **1** | 0.8 |
| Densenet161 | **1** | 0.6 |

|  |  |  |
| --- | --- | --- |
| Model | MIN\_Accuracy (Training)(higher is better) | mIN\_accuracy  (validation) (higher is better) |
| Resnet152 | **0.8666666666666667** | 0 |
| Inceptionv3 | 0.5714285714285714 | 0 |
| exception | 0.3333333333333333 | 0 |
| Densenet161 | 0.6 | 0 |

Metrics

|  |  |  |
| --- | --- | --- |
| Model | F1\_score  (Training) (higher is better) | F1\_score  (validation) (higher is better) |
| Resnet152 | **0.9934928041549645** | **0.20485367075515398** |
| Inceptionv3 | 0.8879849109216635 | 0.1373008546820092 |
| exception | 0.8187631969460836 | 0.14215408247666314 |
| Densenet161 | 0.8827789013863031 | 0.08976616145269721 |

|  |  |  |
| --- | --- | --- |
| Model | Hamming\_loss  (Training) (lower is better) | Hamming\_loss  (validation) (lower is better) |
| Resnet152 | **0.006472491909385114** | **0.7868852459016393** |
| Inceptionv3 | 0.11542610571736785 | 0.8557377049180328 |
| exception | 0.18015102481121897 | 0.8491803278688524 |
| Densenet161 | 0.1186623516720604 | 0.898360655737705 |

|  |  |  |
| --- | --- | --- |
| Model | Jaccard\_score  (Training) (higher is better) | jaccard\_score  (validation) (higher is better) |
| Resnet152 | **0.9875417909099123** | **0.13806384520081713** |
| Inceptionv3 | 0.8091911196298816 | 0.08992359250592097 |
| exception | 0.7065833162696866 | 0.09127099006131263 |
| Densenet161 | 0.7984909354221321 | 0.05340519695358405 |

|  |  |  |
| --- | --- | --- |
| Model | LOG\_loss  (Training) (lower is better) | LOG\_loss  (validation) (lower is better) |
| Resnet152 | **0.024156280111023325** | **5.692915327122016** |
| Inceptionv3 | 0.39418005801246914 | 7.058589225571957 |
| exception | 0.6532672608083165 | 6.8693384256494685 |
| Densenet161 | 0.4880079954175008 | 11.916597437710259 |