# Transfer Learning

An exercise on implementation of Transfer Learning using pretrained Convolutional layers of ResNet50 and CIFAR10 dataset. The dense layer was retrained in the traditional method. Using these large expertly pretrained models gives a huge boost to performance. Specifically, here ResNet50 is 50 layers deep, trained on 1000 different objects, is very apt for the task of image classification in CIFAR10. This can be repurposed for our use through transfer learning where we import pretrained Convolutional layers of ResNet50 and retrain the dense layers on CIFAR10 dataset.

The training time 1h 33s is quite high despite using GPU. Total trainable parameters were 40,583,370 and non-trainable parameters are 315,648. The model has 3 upsampling layers followed by imported resnet50 convolutional layers. Then the tensor is flattened and passed through alternate layers of batch normalized and dense layers for 3 times.

The test accuracy is 92.9% much higher than a 6-layer deep CNN specifically trained on CIFAR10 dataset of 67.6% or any classic ML Models.

Github Link to Notebook:

* CIFAR10 6-layer CNN model <https://github.com/mandalnilabja/soc2022/blob/main/model/Resnet50CIFAR10.h5>
* ResNet50 and CIFAR10 model training notebook: <https://github.com/mandalnilabja/soc2022/blob/main/Week9Assignment.ipynb>

Reference: <https://medium.com/@andrew.dabydeen/transfer-learning-using-resnet50-and-cifar-10-6242ed4b4245>

# Hyperparameter Tuning of CNNpred Implementation

Optimization of performance metrics Accuracy, Mean Absolute Error, Macro Averaged F1 Score by hyperparameter tuning. A Grid-Search strategy for evaluating each hyperparameter combination out of all possible 64 models was adopted.

Following hyperparameters were varied to find the optimal model-

1. Loss Function (mae, binary\_focal\_crossentropy, binary\_crossentropy, hinge)
2. Optimizer (SGD, Adam, Adagrad, Adamax),
3. Epochs (20, 30),
4. Batch Size (64, 32),
5. Dropout Rate (0.05, 0.1, 0.15, 0.2),

The whole experiment took takes 20h+ of training time. And yielded the following results.

CSV containing performance parameters for 64 possible combinations of above hyperparameters-<https://github.com/mandalnilabja/soc2022/blob/main/CNNpred_WallStreet_performances.csv>

Results-

* Optimizers Adam (Adaptive Moment Estimation) and Adagrad is similar in performance but faster than the rest of Optimization algorithms (SGD, Adamax). This is probably due to flatter gradients.
* Initial experimentation was done using batch sizes of 128, 64, 32, 16. Comparing performances revealed 64 and 32 to be better. Thus, these two batch sizes were chosen for further exploration. Finally optimal batch size appeared to be 64 but not by far.
* On average binary\_crossentropy outperforms other loss functions by a small margin
* Dropout rate has no significant effect on evaluation metrics
* Training for 30 epochs and batch sizes 64 overfit the data with very low incremental validation loss but low F1 Score

Github Link to Notebook:

<https://github.com/mandalnilabja/soc2022/blob/main/Week9AssignmentB.ipynb>

# Ensemble- Bagging

Bagging ensemble on CNNpred implementation using all 64 previous models outperformed any single predictor. A hard voting strategy was manually implemented since each model by itself is a weak classifier.

Its evaluation metrics were:

Accuracy: 53.75%

Mean Absolute Error: 46.24%

Macro Averaged F1 Score: 69.92%

Github Link to Notebook: <https://github.com/mandalnilabja/soc2022/blob/main/Week9AssignmentC.ipynb>

Models: <https://github.com/mandalnilabja/soc2022/tree/main/model>