**Apply fine tuning for your team’s network to the CIFAR 10 dataset.**

Team’s network: DenseNet169

**Fine tuning** is a technique to use pre-trained models and how to extract features from them for training a model for a different task.

In fine tuning we freeze the initial layers and retain the later layers for our task.As [explained here](https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/#cnn-hierachical), the initial layers learn very general features and as we go higher up the network, the layers tend to learn patterns more specific to the task it is being trained on.

In addition to the base DenseNet169 a dense layer and a softmax were added to classify the output.

**Experiments performed:**

1. **Training only the last convolutional layer:**

**Results:** Best test accuracy and test loss obtained in this method (using various dropouts) was 0.4408 and 2.7471.

1. **Training the last two convolutional layers:**

**Results:** Best test accuracy and test loss obtained were 0.4672 and 2.8262 respectively.

1. **Training the last three convolutional layers:**

Results: An accuracy of 0.4296 and a loss of 2.8412 were observed.

Results from other experiments were listed below.

**Experiments:**

**Base Model:** Trainable layers: base\_model.layers[:-4]

## Unfreeze the last conv layer, pooling and norm layers.

Base model: model.compile(loss='categorical\_crossentropy',

optimizer = optimizers.Adam(lr=1e-4), metrics=['acc'])

***Model 1:***

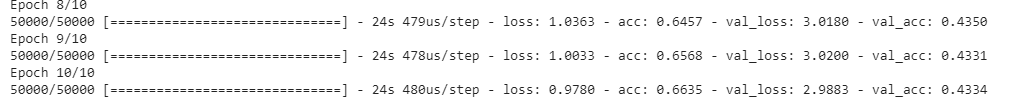
Added: ##Add new Dense layer and a softmax to classify.

model.add(layers.Flatten())

model.add(layers.Dense(1024, activation = 'relu'))

model.add(layers.Dense(10,activation='softmax'))

**Result after 10 epochs:**



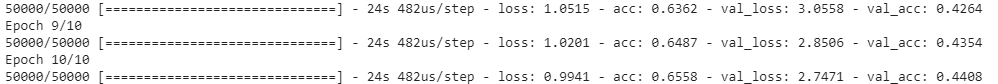
***Model 2:***

model.add(layers.Flatten())

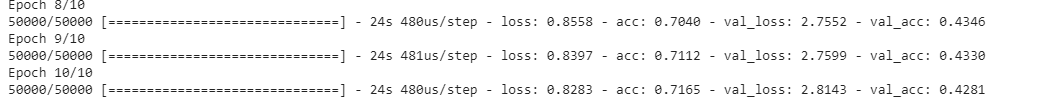
model.add(layers.Dense(1024, activation = 'relu'))

model.add(layers.Dropout(0.2)) **##Added a dropout to base model**

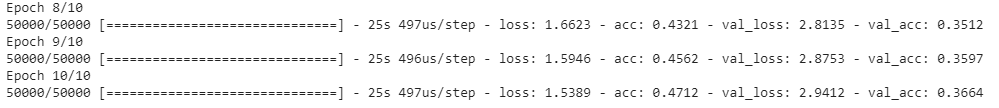
model.add(layers.Dense(10,activation='softmax'))



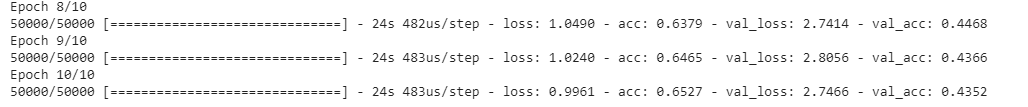
**Model 3:** All same as model 2 but changed Adam to **RMSprop**



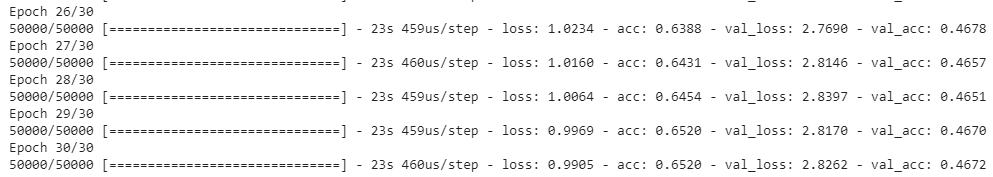
**Model 4:** All same as base but learning rate is **1e-5, base\_model.layers[:-8] ##Unfreeze the last 2 conv2D layers of base model**



**Model 5:** Same as 3 with **dropout 0.3**



**Model 6:** All same as base but learning rate is **1e-5, base\_model.layers[:-7]** and 30 epochs



**Model 7:** All same as base with unfreeze the last 3 conv layers: