Assignment 3

Transfer Learning

Hari Krishnan CK

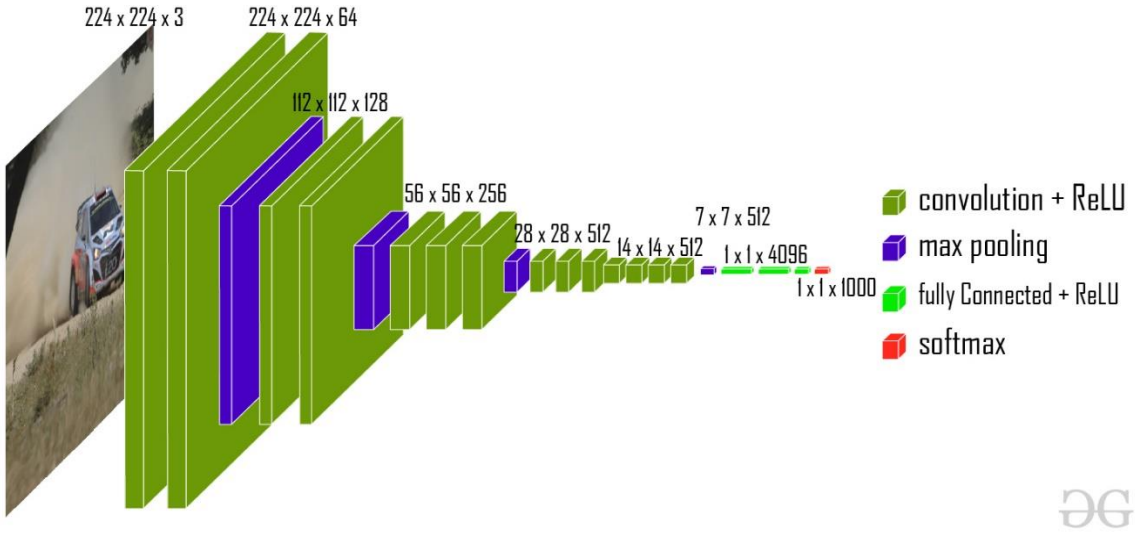
2021EEY7583

Aim: Transfer learning on an action dataset using the pre-trained weights of VGGNet trained on ImageNet dataset.

Dataset: Dataset contains 3411 training images of size (224X224X3) and 125 testing and validation images of same size. There are 25 classes in the data so the last layer ouput of our classifier network should be 25.

Methodology: Pretrained VGGNet was taken from torchvision models and the last classification layers are replaced with a new classifier architecture. The feature extraction part was kept frozen and not trained.

VGGNet Architecture:



We keep the CNN feature extraction part which gives output of dimension 7X7X512. These outputs are passed through a fully connected classifier which gives the probabilities for our dataset.An example of the classifier architecture is given below.

```
(fc): Sequential(
  (0): Linear(in_features=25088, out_features=4096, bias=True)
  (1): ReLU(inplace=True)
  (2): Linear(in_features=4096, out_features=4096, bias=True)
  (3): ReLU(inplace=True)
  (4): Linear(in_features=4096, out_features=4096, bias=True)
  (5): ReLU(inplace=True)
  (6): Linear(in_features=4096, out_features=25, bias=True)
)
```

Experiments:

Please see the resultsSummary.csv for the complete statistics.

Base line classifier architecture with no dropout and neurons [25088,4096,4096,4096,25]:

Data augmentation used: Only resize and normalization with mean (0.485, 0.456, 0.406) and standard deviation (0.229, 0.224, 0.225)

```
  (fc): Sequential(
(0): Linear(in_features=25088, out_features=4096, bias=True)
(1): ReLU(inplace=True)
(2): Linear(in_features=4096, out_features=4096, bias=True)
(3): ReLU(inplace=True)
(4): Linear(in_features=4096, out_features=4096, bias=True)
(5): ReLU(inplace=True)
(6): Linear(in_features=4096, out_features=25, bias=True)  )
```

Epochs run :25

Optimizer used: SGD with lr=0.01 and momentum=0.8.

Scheduler: MultiStepLR with steps at [3,10,15]

Best validation accuracy: 99.2%

Training accuracy obtained: 94.4%

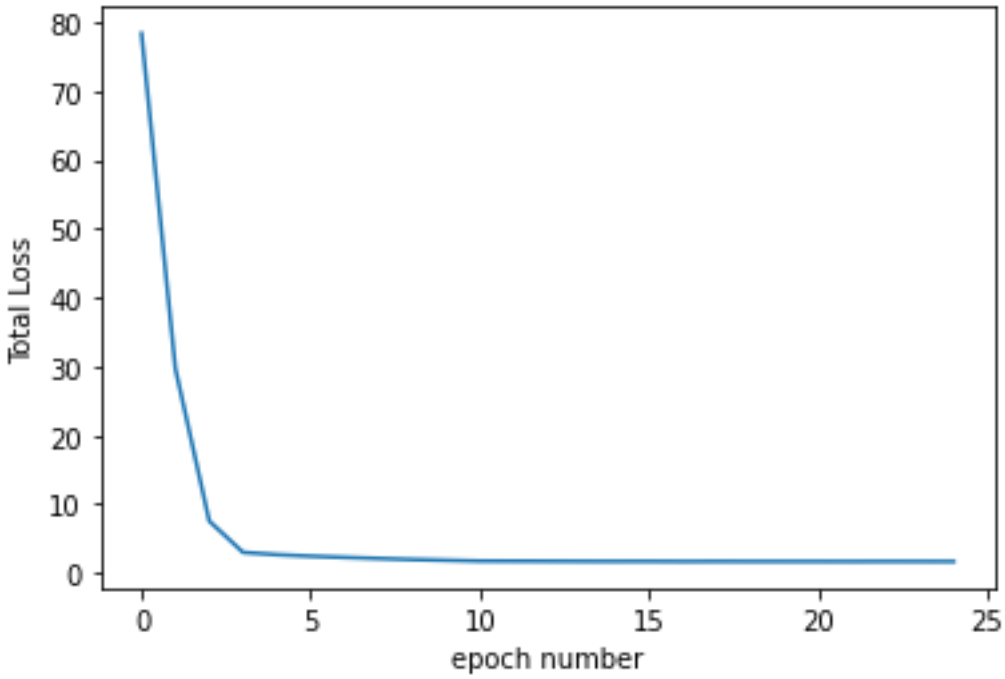


Figure: Training loss vs epoch number

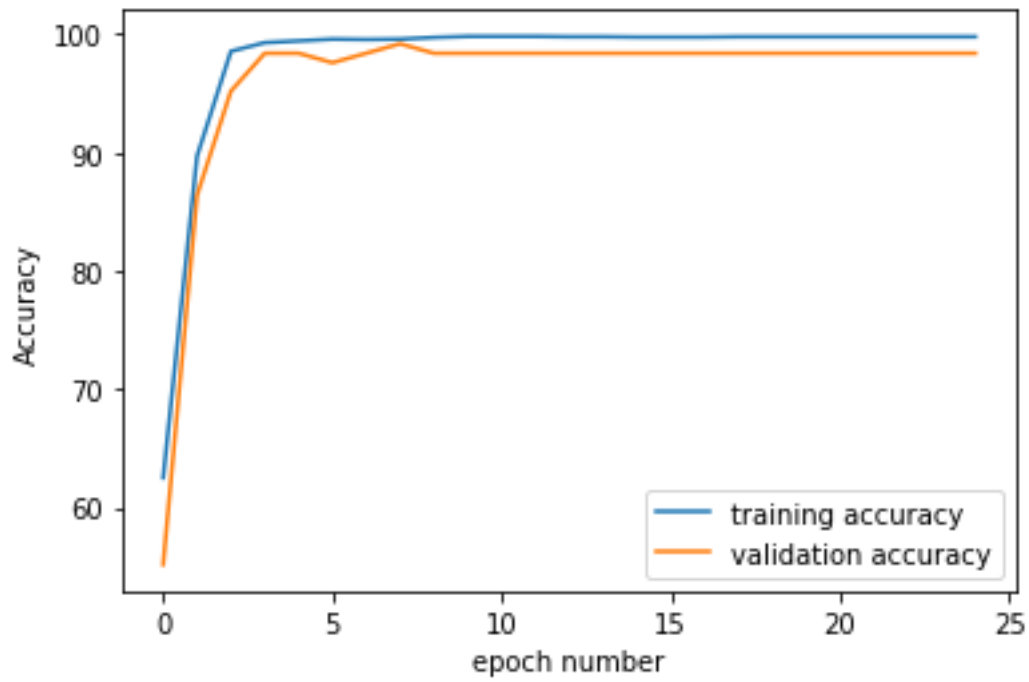


Figure: Validation vs Training accuracy

The above network will be used as a Base line to study the various regularisation techniques.The list of all experiments performed is provided in the csv. Screenshot of the same is given below.

Experiment No	Data Augmentation	VGGNet trained or not	Classifier architecture	Activation function	Dropout	BN	Optimizer	Scheduler	l2-l1	Best validation accuracy	Test accuracy
1	No	No	25088-4096-4096-25	Relu	No	No	SGD lr=0.01	Step 3-10-15	No	98.4	94.4
2	No	No	25088-4096-4096-25	Tanh	No	No	SGD lr=0.01	Step 3-10-15	No	99.2	97.6
3	No	No	25088-4096-4096-25	Relu	Yes 0.5-0-0	No	SGD lr=0.01	Step 3-10-15	No	98.4	93.6
4	No	No	25088-4096-4096-25	Relu	Yes 0.25-0-0	No	SGD lr=0.01	Step 3-10-15	No	99.2	96
5	No	No	25088-4096-4096-25	Relu	Yes 0.1-0-0	No	SGD lr=0.01	Step 3-10-15	No	99.2	96
6	No	No	25088-4096-4096-25	Tanh	Yes 0.1-0-0	No	SGD lr=0.01	Step 3-10-15	No	99.2	97.6
7	No	No	25088-15000-5000-2500	Relu	No	No	SGD lr=0.01	Step 3-10-15	No	98.4	96.8
8	No	No	25088-15000-5000-2500	Relu	Yes 0.1-0-0	No	SGD lr=0.01	Step 3-10-15	No	98.4	96.8
9	No	No	25088-15000-5000-2500	Tanh	Yes 0.1-0-0	No	SGD lr=0.01	Step 3-10-15	No	98.4	96
10	Yes Rotation	No	25088-4096-4096-25	Relu	No	No	SGD lr=0.01	Step 3-10-15	No	99.2	98.4
11	Yes Rotation	No	25088-4096-4096-25	Tanh	Yes 0.1-0-0	No	SGD lr=0.01	Step 3-10-15	No	99.2	97.6
12	Yes Rotation, RandomCrop	No	25088-4096-4096-25	Relu	No	No	SGD lr=0.01	Step 3-10-15	No	98.4	94.4
13	Yes Rotation	No	25088-4096-4096-25	Relu	No	Yes	SGD lr=0.01	Step 3-10-15	No	98.4	96.8
14	No	No	25088-4096-4096-25	Relu	No	Yes	SGD lr=0.01	Step 3-10-15	No	99.2	97.6
15	No	No	25088-4096-4096-25	Relu	No	No	SGD lr=0.1 m	Step 3-10-15	No	100	96.8
16	No	No	25088-4096-4096-25	Relu	Yes 0.1-0.1-0.1	No	SGD lr=0.01	Step 3-10-15	No	99.2	96.8
17	No	No	25088-4096-4096-25	Relu	Yes 0.1-0-0	Yes	SGD lr=0.01	Step 3-10-15	No	99.2	97.6
18	Yes	No	25088-4096-4096-25	Relu	Yes 0.1-0-0	Yes	SGD lr=0.01	Step 3-10-15	No	98.4	94.4
19	No	No	25088-4096-4096-25	Relu	No	Yes All layers	SGD lr=0.01	Step 3-10-15	No	98.4	97.6
20	No	No	25088-4096-4096-25	Relu	No	No	SGD lr=0.01	Step 10-17-21	l1 after 6 epo	99.2	96
21	No	No	25088-4096-4096-25	Relu	No	No	SGD lr=0.01	Step 10-17-21	l2	99.2	96.8
22	No	No	25088-4096-4096-25	Relu	No	No	SGD lr=0.01	Step 12-15-17	l1 after 2 epo	100	96.8
23	No	No	25088-4096-4096-25	ReLu	No	No	ADAM lr=0.0	Step 3-10-15	No	99.2	90.4

Figure: Experiments performed

a.) Activation function: We ran the same architecture for the classifier with the only difference being the activation function used. Tanh and ReLu was tried and in this dataset with no dropout and batch normalization Tanh tends to perform better than ReLU.

With Tanh test accuracy obtained: 98.4%

With ReLU test accuracy obtained: 94.4%

So using Tanh instead of ReLU tend to perform slightly better.

b.) Dropout : The effect of dropout on the accuracy is tested in this section. We add dropout on only the first layer only (25088-4096 layer) to check the performance. On the network with ReLU activation function:
    a. p=0.5 : In this case 50% of the neurons from the first layer were dropped. We observed a dip in performance from 94.4% to 93.6%.
    b. p=0.25 : Here only 25% of the neurons were dropped and an increase in performance to 96% from 94.4% was observed.
    c. P=0.1 : Here also same increase as in p=0.25 was observed

    From this we can observe that the dropout hyperparameter should be tuned for good performance as it can also decrease the accuracy if it is too high. In the same way when we performed on the network with Tanh activation function no change in accuracy was observed.

c.) Dropout in all layers: The effect of dropout on all layers on the classifier was also studied. The observation was that the accuracy increased from 96% to 96.8% when using single layer to multiple dropout layers.

d.) Number of neurons: Now we try to increase the number of neurons in each layer in the baseline model. We increase the neurons in each layer i.e the architecture changes from(25088-4096-4096-4096-25) to (25088-15000-5000-2500). We observed a significant increase in accuracy from 94.4% to 96.8% in this case. However it is not always true since when we increased the architecture with tanh activation function the accuracy of the model actually dipped from 97.6% to 96%. Hence it is not always true that increasing the complexity will increase the accuracy.

e.) Depth of the network: In this section the depth of the network was increased from (25088-4096-4096-25) to (25088-4096-4096-4096-4096-25). We observed a decrease in accuracy.

f.) Data Augmentation: Here we augment the data to increase the variance in the input data.
    a. RandomRotation of 20 degrees: The baseline model architecture was used. We observed a significant increase of 4% from 94.4% to 98.8% with just random rotation of 20 degrees as data augmentation. Although for Tanh network the accuracy remained constant.
    b. RandomRotation and RandomCrop: To the above data augmentation technique, random crop of 150X150 was also added. This surprisingly reduced the accuracy from 98.4% to 94.4% only performing as good as the baseline model only.

    This shows that all data augmentation techniques may not be good for the model and we have to choose the data augmentation techniques very carefully so that It works well in our case.

g.) Batch Normalization: Batch normalization is another very efficient way to reduce the internal covariance shift.
    a. Batch Normalization only on the first layer (25088-4096): In this case we observed an increase in accuracy from 94.4% to 97.6%.
    b. Batch Normalization on all the layers in the classifier: BN on all the layers on the classifier did not give any improvement from the current 97.6%.

h.) Dropout and Batch Normalization together: Batch normalization together with dropout on only the first layer gave the expected result of 97.6% which was the maximum accuracy when only batch normalization was used alone.

i.) Dropout, BN and data augmentation together: In this case we used the techniques like dropout, batch normalization and data augmentation

j.) Learning rate: In this section, we try to change the learning rate to study the effects of changing the learning rate. We changed the starting learning rate fro 0.01 to 0.1. This will effectively change the learning rate since we are changing the lr after steps of [3,10,15]. The main observation here was that the validation data accuracy reached 100%. And the testing data accuracy increased to 96.8%.

k.) L1 regularisation: L1 regularisation on the weights of the classifier was also done on the base model. Implemented in pytorch as “”sum(torch.linalg.norm(p, 2) for p in model.fc.parameters())””

   a. L1 after 6 epochs: Here we apply L1 after 6 epochs of optimising with crossentropy is done.L1 relugarisation tend to help the model as we got an increase in accuracy from 94.4% to 96%.
   b. L1 after 2 epochs: Here we apply L1 regularisation very early into the training i.e only after 2 epochs. The accuracy obtained was increased to 96.8% compared to later L1.

l.) L2 regularisation: L2 regularisation on the weights of the classifier also gave an increase in accuracy to 96.8%.

m.) Optimizer: When we changed the optimizer from SGD to ADAM keeping the learning rate fixed, the accuracy dropped to 90.4% from 94.4%. We may have to tune the learning rate specifically for ADAM.

Conclusions

a.) The choice of activation function affects the performance of the model. In this case tanh tends to work slightly better than ReLU.

b.) Dropout is a form of L2 regularisation which prevents overfitting by averaging the model. However if p value in dropout is very high, then it will lead to a decrease in performance.

c.) Network architecture also plays an important role since if it is highly complex, then the model performs very badly on the test data and if it is too small no learning may happen at all.

d.) Data augmentation should increase the performance most of the times but as it is observed in this case adding random crop didn’t improve the accuracy rather accuracy was reduced. But we also got the highest accuracy when we used randomRotations of 20 degrees on the training data

e.) Batch normalization also increased the accuracy when it was applied on the base line. BN tries to take care of internal covariance shift by subtracting the mean and standard deviation. Although applying them in multiple layers didn’t improve results in this case.

f.) L1 and L2 regularisation on the weights is also a good regularisation technique and it showed some improvement. Although the importance or weight to be given to these terms in loss function should also be tuned.

g.) Optimizer: Choice of optimiser is another important hyperparameter that we have to choose. We tried with ADAM along with SGD with the same learning rate. Although it was expected that ADAM will work better, in this case actually ADAM performed significantly worse than SGD with a drop of approximately 4% in accuracy.

h.) Applying these techniques together may not always give the best results as it is shown in the experiments conducted. Some cases we may need to apply only dropout or data augmentation and there may not be a need for other techniques and applying these may reduce the performance.s