BBM 418 2_{nd} Assignment Report

Leyza Yortanlı
21328634

April 28, 2018

Abstract

In this assignment we tasked with the do transfer learning on VGG-16 that trained with ImageNet. Aim is to analyze and observe the results, the effects of the parameters etc. In this paper there will be the explanation of whole concept of fine tuning, details about implementation, the techniques to get better results, the models and the further extensions.In the end there will be the analysis section to share the observed results from the model that fine tuned. And also we will compare the different models in each other also.

1 Introduction

First of all Warren McCulloch and mathematician Walter Pitts wrote a paper on how neurons might work then after years computer scientist attempt to simulate same behaviours on computers.They went step by step, first implemented perceptrons then they increase the neuron, the layers.. Then neural network is improved by the years.Convolutional Neural Networks are very similar to ordinary Neural Networks.A convolutional neural network (CNN, or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery.
In this assignment we focused on this kind of neural network and analyzed specialized models for this networks.Let's give a brief summary about the cnn and models that we have used.

2 Convolutional Neural Networks

Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores.[1] As this kind of structure does not scale for the full image Convolutional neural networks comes out. CNN inputs are has a volume compare to regular neural networks.Which makes it easier to handle image data. This volumes does not represents to layers but represents the inputs.

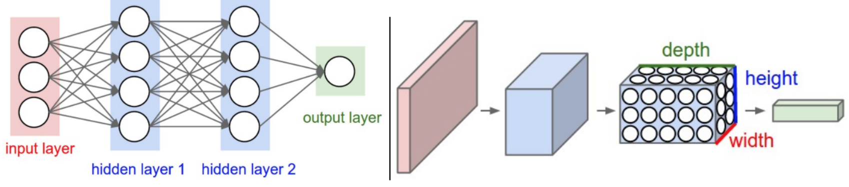


Figure 1: $NN - CNN_{[1]}$

Every layer of CNN transforms one volume of activations to another through a differentiable function. There is three main types of layer in CNNs:

- Convolutional Layer : Computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.[1] This actually kind of doing filtering operation.

- Pooling Layer : Doing downsampling operation for prevent the overfitting.
- Fully-Connected Layer : Computes the class scores.

And also between the layers there is an activation functions which is a way of interpreting the results.Every activation function takes a single number and performs a certain fixed mathematical operation on it.As a summary these kind of functions decides whether neuron should be fired or not.
In CNN input is kind of common compare to other neural networks. As CNN handles the images and in the end vision of the world will always have a common parts. So in my dream one day there will be one and whole cnn model that will be used for the all kind of recognition, description jobs. As this is a deep and long story let's directly go with the models that are based on CNN.

3 Models

There is too many architecture based on CNN.According to the universal approximation theorem, given enough capacity, we know that a feedforward network with a single layer is sufficient to represent any function. However, the layer might be massive and the network is prone to overfitting the data. Therefore, there is a common trend in the research community that our network architecture needs to go deeper.[3]So this models appeared. They're all better than another to some cases. So we should know their characteristics to use them better.Of course we cannot explain all of them here, let's go with the model that I use for this assignment.

3.1 VGG-16

VGG is stands for Visual Geometry Group which is the inventor group name of this architecture.

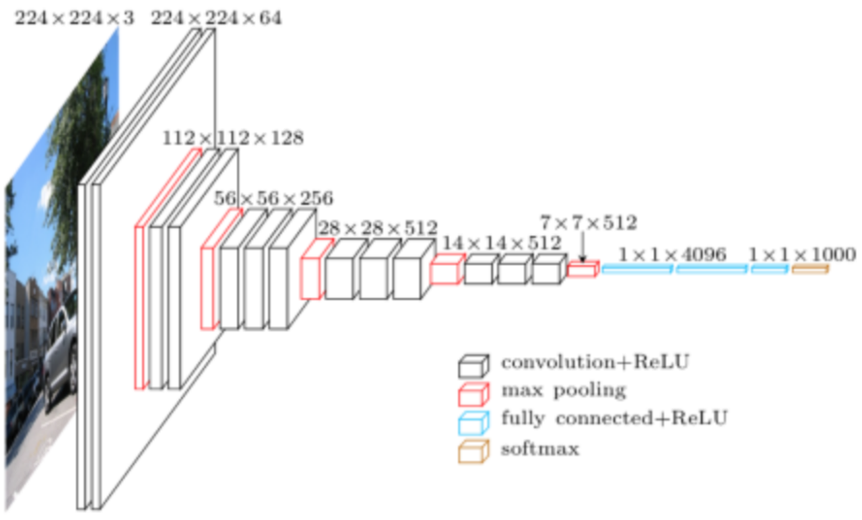


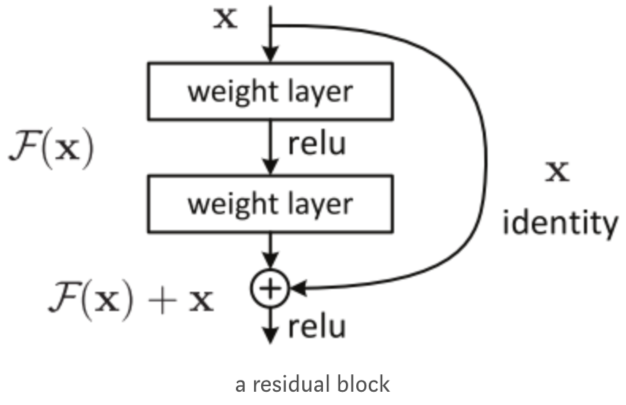
Figure 2: VGGNet

As I am not qualified enough to explain the architecture of VGGNet I am directly quoting their explanation below.
"During training, the input to our ConvNets is a fixed-size 224 × 224 RGB image. The only preprocessing we do is subtracting the mean RGB value, computed on the training set, from each pixel. The image is passed through a stack of convolutional (conv.) layers, where we use filters with a very small receptive field: 3 × 3 (which is the smallest size to capture the notion of left/right, up/down, center). In one of the configurations we also utilise 1 × 1 convolution filters, which can be seen as a linear transformation of the input channels (followed by non-linearity). The convolution stride is fixed to 1 pixel; the spatial padding of conv. layer input is such that the spatial resolution is preserved after convolution, i.e. the padding is 1 pixel for 3 × 3 conv. layers. Spatial pooling is carried out by five max-pooling layers, which follow some of the conv. layers (not

all the conv. layers are followed by max-pooling). Max-pooling is performed over a 2 $\times$ 2 pixel window, with stride 2. A stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000- way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer. The configuration of the fully connected layers is the same in all networks"[2] This model won the ImageNet Competition in 2015 in the localisation and classification categories. So as this model already learned a lot, it is too usefull to do transfer learning.

3.2 Resnet18

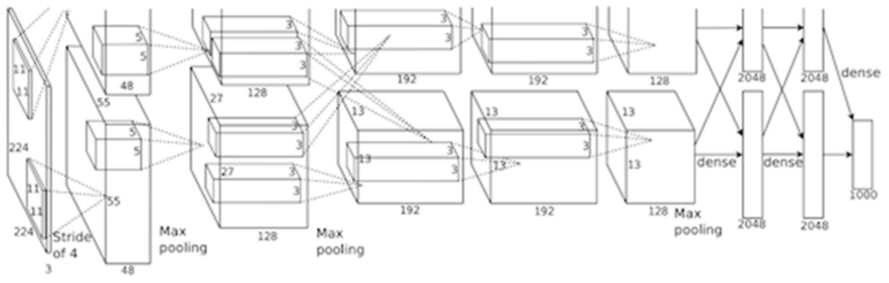
Resnet is stands for "Deep Residual Networks For Image Recognition".ResNet makes it possible to train up to hundreds or even thousands of layers and still achieves compelling performance.The core idea of ResNet is introducing a so-called "identity shortcut connection" that skips one or more layers, as shown in the following figure:[3]


x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu
a residual block

They explain the architecture features in corresponding paper like following: "..Mainly inspired by the philosophy of VGG nets.The convolutional layers mostly have 3×3 filters and follow two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. We perform downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax. It is worth noticing that our model has fewer filters and lower complexity than VGG nets. Our 34- layer baseline has 3.6 billion FLOPs (multiply-adds), which is only 18% of VGG-19 (19.6 billion FLOPs)."[4]

3.3 AlexNet

This is like the fathers of above. This architecture is the one who started all these stories.Originally written with CUDA to run with GPU support,competed in the ImageNet Large Scale Visual Recognition Challenge in 2012. The network achieved a top-5 error of 15.3%, more than 10.8 percentage points ahead of the runner up. AlexNet was designed by the SuperVision group, consisting of Alex Krizhevsky, Geoffrey Hinton, and Ilya Sutskever.[5] This architecture relatively simple than the others.Made up of 5 conv layers, max-pooling layers, dropout layers, and 3 fully connected layers



AlexNet architecture (May look weird because there are two different "streams". This is because the training process was so computationally expensive that they had to split the training onto 2 GPUs)

This network is first successful CNN application for such a big dataset(Imagenet). They also do data augmentation to prevent overfitting which will be explain under tecniques section.

4 Transfer Learning

Transfer Learning is an approach to use pretrained model for different task. It can be think like two phase for training. First phase is completed with large datasets, for the small datasets like we have we can take this model as a starting point and retrained not from the stratch but from the middle or after the feature extraction part.It simply teaching some other category images to model that it hasn't seen before.It is simply multi tasking solver approach.
There is two common approach to transfer learning:

1. Develop Model Approach
   (a) Select Source Task. You must select a related predictive modeling problem with an abundance of data where there is some relationship in the input data, output data, and/or concepts learned during the mapping from input to output data.
   (b) Develop Source Model. Next, you must develop a skillful model for this first task. The model must be better than a naive model to ensure that some feature learning has been performed.
   (c) Reuse Model. The model fit on the source task can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
   (d) Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.
2. Pre-trained Model Approach
   (a) Select Source Model. A pre-trained source model is chosen from available models. Many research institutions release models on large and challenging datasets that may be included in the pool of candidate models from which to choose from.
   (b) Reuse Model. The model pre-trained model can then be used as the starting point for a model on the second task of interest. This may involve using all or parts of the model, depending on the modeling technique used.
   (c) Tune Model. Optionally, the model may need to be adapted or refined on the input-output pair data available for the task of interest.

·[6]
In this assignment we followed the mostly using approach in these kind of models which is second approach. We used the models that pre-trained for a large and challenging image classification task such as the ImageNet.As it would be nearly impossible for us to train this kind of networks in this kind of dataset we use the knowledge that model has from the these datasets. And we add new tasks to be learn for the model.By following this approach we win over following issues:

1. Higher start. The initial skill (before refining the model) on the source model is higher than it otherwise would be.

2. Higher slope. The rate of improvement of skill during training of the source model is steeper than it otherwise would be.
3. Higher asymptote. The converged skill of the trained model is better than it otherwise would be.

·[6]
If we would summarize transfer learning is a way of creating the solution for the new problems by using the knowledge that has learned before. To have a good model there is some techniques to prevent overfit - underfit or reach a better accuracy results. Next section we will discuss about some of those tecniques that we used.

4.1 Tecniques

As not every training is end up with good results in this section we will discuss about general problems and what causes them. After that we will briefly discuss the solutions to those problems. When training a model it is important that model generalizes the data as data is not complete and noisy.If this not happened probably you faced with one of the two main problem which are overfitting and underfitting.
In statistics, a fit refers to how well you approximate a target function.

- Overfitting :Refers to a model that models the training data too well.Overfitting happens when a model learns the detail and noise in the training data to the extent that it negatively impacts the performance of the model on new data. This means that the noise or random fluctuations in the training data is picked up and learned as concepts by the model. The problem is that these concepts do not apply to new data and negatively impact the models ability to generalize.
- Underfitting : Underfitting refers to a model that can neither model the training data nor generalize to new data.An underfit machine learning model is not a suitable model and will be obvious as it will have poor performance on the training data. Underfitting is often not discussed as it is easy to detect given a good performance metric. The remedy is to move on and try alternate machine learning algorithms. Nevertheless, it does provide a good contrast to the problem of overfitting.[8]

4.1.1 Shuffling

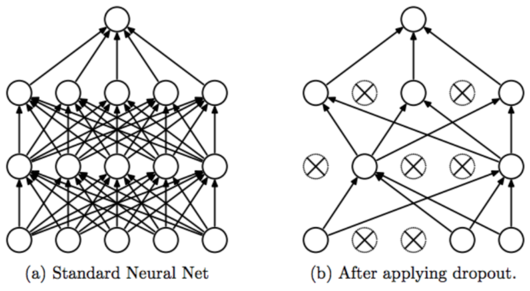
Shuffling is just a good approach that works with mini batches. When you shuffle the images you decrease the number of images which are in same label that will come in same batch.So you make more mixed each batch. So it is a way of prevention from underfit or overfit for only one class. It is intention to make more generalize the model.

4.1.2 Data Augmentation

As most of the time for image and video classification problems has a insufficient data, this tecniques proposed(paper :[9]).In that paper there is two approach to increase the data first is traditional data transformation which consist of using a combination of affine transformations to manipulate the training data. And the second approach is Generative Adversarial Networks which is doing for each input image, we select a style image from a subset of 6 different styles and do transformation of the original image is generated.In that paper by this tecniques they proved it increased the accuracy.Trade off for this tecnique is you need to allocate more memory.

4.1.3 Dropout

The term dropout refers to dropping out units (both hidden and visible) in a neural network.It simply ignoring randomly selected neurons in training phase to prevent from overfitting otherwise it might be focused on some neurons and do hard performance on them and can end up with overfitted results.



4.2 Batchsize and Epoch

In training phase we want to minimize the loss and maximize the accuracy. Because of that we use gradient descent algorithm to reach minimum loss.Gradients means the rate of inclination or declination of a slope. When you're moving on this slope you might need to move on it more than once. So iteration comes from this idea.The iterative quality of the gradient descent helps a under-fitted graph to make the graph fit optimally to the data.Because of that we do multiple epochs as the passing the entire dataset through a neural network is not enough and we need to pass the full dataset multiple times.As this is a data dependent choice there is no perfect number for epochs.
And because of memory problems we can't feed all images at once, so we divide it in multiple batches.And as we iterate on batches to network is tend to train faster as it update the weights for each batches.Only disadvantage for this tecnique is the smaller batches creates less accurate gradients. So it balanced harder.You can see the example of results in picture belove.

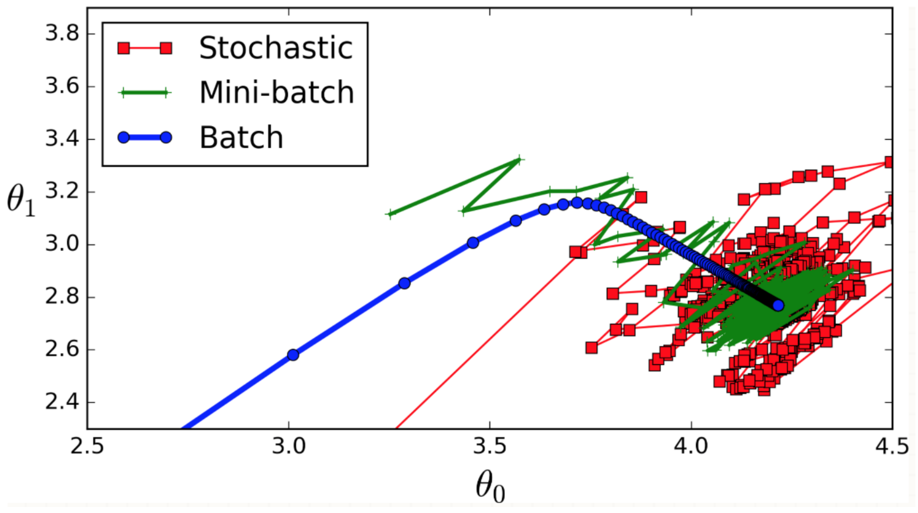


Figure 3: Image is taken from :[10]

5 Implementation

In my implementation I used PyTorch, and I used the pretrained models for VGG16 and ResNet18.I followed the instructors of transfer learning that is given in documentation of library.
As pytorch has a imagefolder functionality which is a library for understand the labels all the images and creates iterable data.To use that functionality we seperate the images into train and test folders with respect to given txt files.Then with giving just the directory names and it handle the rest. Than we chose following functions for training phase :

1. Criterion : Chose Cross Entropy Loss as it has softmax loss calculation in it and softmax is the right loss function for multiclass classification problems.
2. Optimizer : Chose stochastic gradient descent as it is efficient according to implementation and running.It is proven that good for large scale machine learning problems.

3. Scheduler : Used StepLR.It is a decay function which is do following according to do documentation : "Sets the learning rate of each parameter group to the initial lr decayed by gamma".

Only difference between the implementation of three model is that vgg16 does not have a fc variable. In their implementation they hold the fully connected layer in classifiers sequential set. So because of that implementation has done according to that.
Last but not least, I implemented top5 accuracy by the help of topk function which implementation found in pytorch forum website.[11]
As a comment I can say that pytorch has all the functionality for these kind of problems.So I cannot say that I created the algorithm or offer a solution in implemantation wise. I only use the building blocks of corresponding library.

6 Results and Analysis

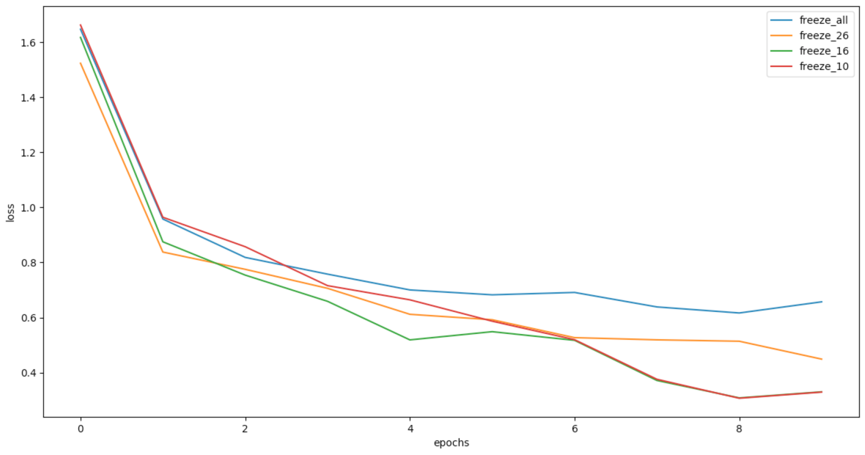
In this assignment I changed a few parameters to see their affects on model training and results. In this section we will discuss them accordingly the information we explained above.

6.1 Freezed Layer Numbers

When I print out the parameters in VGG16, see the following :

features.0.weight	features.10.weight	features.19.weight	features.28.weight
features.0.bias	features.10.bias	features.19.bias	features.28.bias
features.2.weight	features.12.weight	features.21.weight	classifier.0.weight
features.2.bias	features.12.bias	features.21.bias	classifier.0.bias
features.5.weight	features.14.weight	features.24.weight	classifier.3.weight
features.5.bias	features.14.bias	features.24.bias	classifier.3.bias
features.7.weight	features.17.weight	features.26.weight	classifier.6.weight
features.7.bias	features.17.bias	features.26.bias	classifier.6.bias

So I tried freeze them all and only train last classifier and then I unfreezeed the layers step by step. In this experiment we set other parameters as : batch = 16, epoch= 10, lr = 0.001.



In here freeze all means obviously freeze them all, and $freeze\_x$ means that freeze first x parameters above.
As you can see from results model is tend to be better when we freeze less layer. The reason behind that is this model is trained for the imagenet dataset, from the beginning parts it is mostly about feature extracting but when it reach the last layer it becoming specialized for the dataset.As feature extraction is learned perfectly we don't actually need to freeze first 3-4 layers.When we unfreezed the layers after them, it becomes more specialized for our dataset so loss decreases more than others.

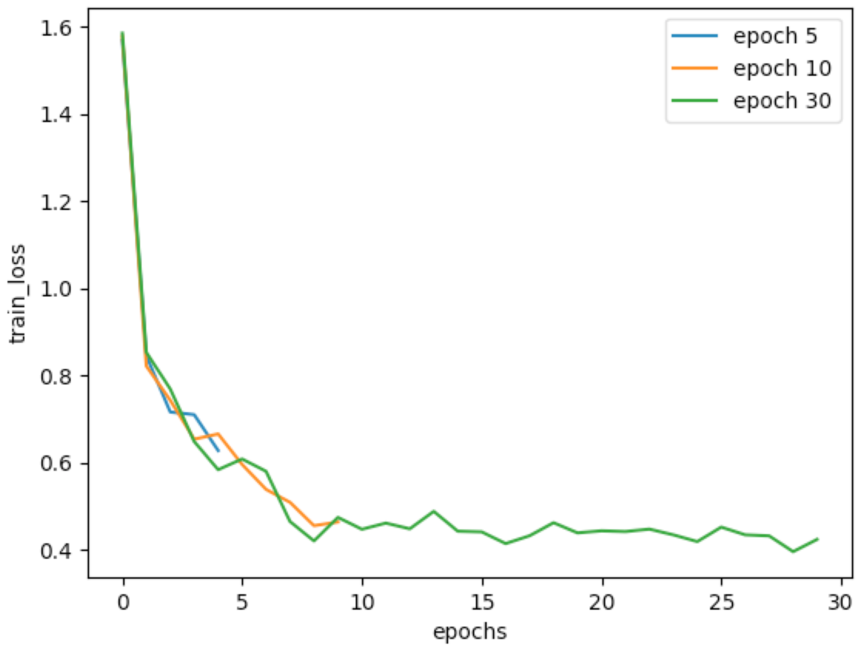
But unfreezing more layers does not mean that it will be always decrease the loss.It is not changing after some point(look the freeze_16 and freeze_10) because it again learns the extract the same features so it does not need re-train for those layers.Unfreezing more than some point is just a time consuming job and at the same time it might be something mislead as we don't want to retrain the model we want to fine tune it. So for perfect fine tuning unfreezed layers should choose wisely.

6.2 Number Of Epochs

For this experiment I set the batch size as 16, learning rate(lr) as 0.001, and freezed first 13 layers.And I tried the training for 5, 10 and 30 epochs. I wanted to train for more than 30 but for some computational issues I could not be able to train that long. To compare this parameter I decided to take the best test accuracy scores.I observed the aproximately results like following :

1. For 5 epoch : 0.81
2. For 10 epoch : 0.84
3. For 30 epoch : 0.85

As you can see increasing the epochs increased the accuracy. I wish I would be able to see the point that in converges but unfortunately I could not be able to see it. But heuristically I would say that it would be converged at the 30-50 epochs.
And epoch is not all about the reaching the best accuracy. Also we can observe the stability of the model by interpreting epoch loss line's slopes.



It might seem lines are following each others direction but the point is not that. For few epoch you can see that line is not stabilized so training should keep going. But the 30 epoch nearly enough to say that loss in stabilized in some point which seems nearly converged. As I said above I still did not see the perfect converged lines as I could not be able to run more than 30 epoch in my computer.

Other than that we can observe epoch effects on training by comparing the top1 and top5 errors for each epoch:

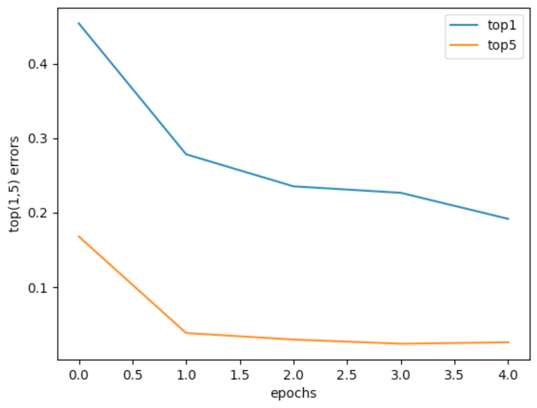


Figure 4: 5 epoch

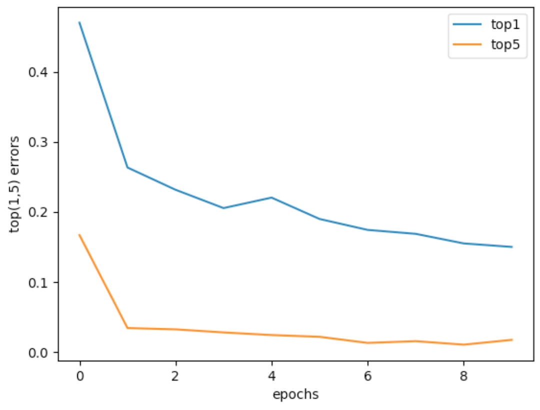


Figure 5: 10 epoch

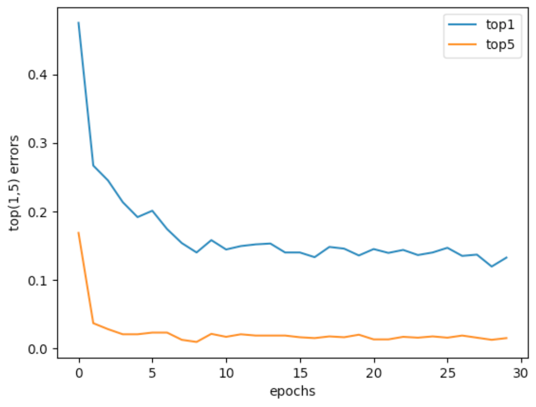
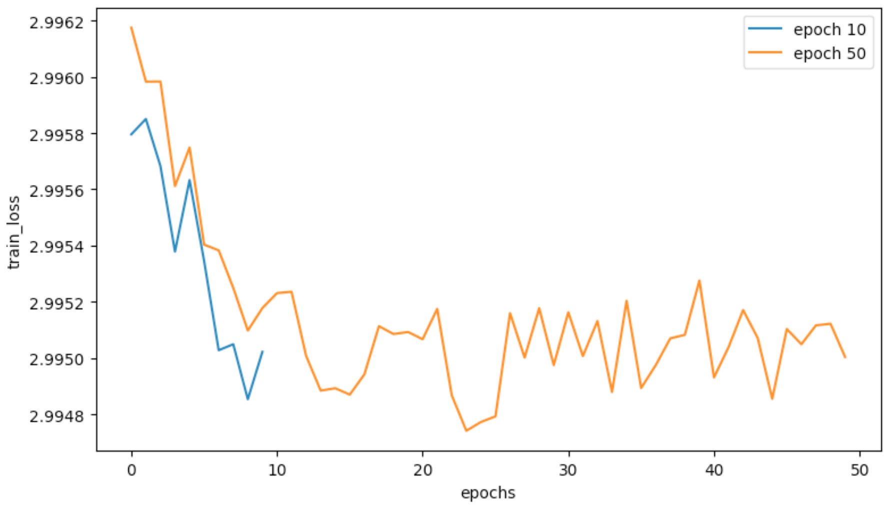


Figure 6: 30 epoch

As you can slightly see top1 and top5 lines are approach to each other when epoch in increased.Probably if we could manage we would have seen that they get closer with more epochs so then we can reach the best model for this dataset.

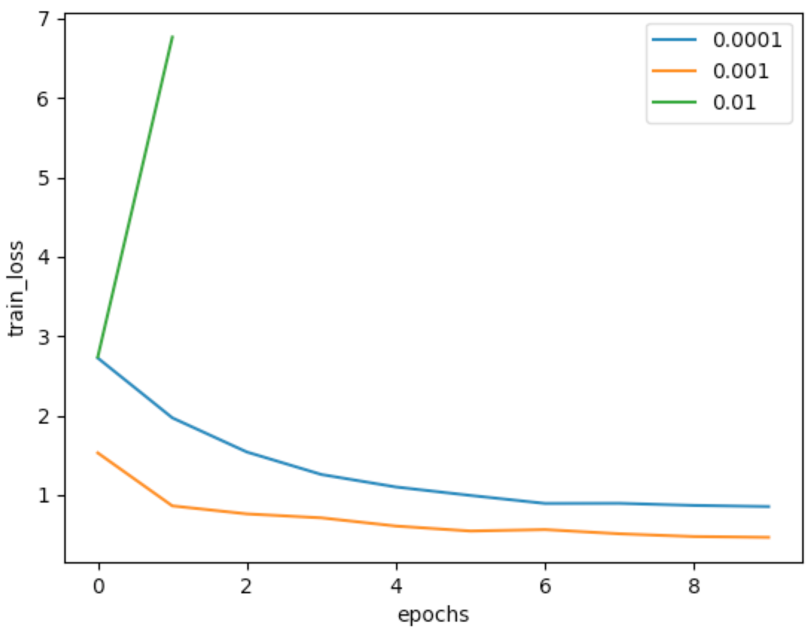
And in addition to top1 and top5 errors let's see those scores in reverse version which is top1 and top5 errors:



I also want to point that number of epoch is a parameter that is highly dependent to model architecture too. As AlexNet has not so deeper architecture I was able to run 50 epoch. But as you can see for even 50 epoch it is not converged even for the same dataset. So that is clearly show a reason why fine tuning is some times more beneficial than training from stracth. With less epoch you are able to get the better results.

6.3 Learning Rate

For this experiment we wanted to observe the learning rate effects on training so for this experiment we set the parameters like following : epoch = 10, batch = 16, freezed first 13 layers.



In addition to that table I tried also 0.1 for learning rate but I could not be able to observe any loss results. Even for 0.01 after two epoch it overflows and loss computation returns None. In the same time as we can see above decreasing learning rate does not mean that it will find the right point better it makes process slower. For this experiment I observed that 0.001 works fine. It is converges faster and it is able to find optimal point.

Of course this is not the right way for choosing learning rate but in limited time and computer power I only observed this much. But let me share this image belove to see the right choosing way of learning rate.

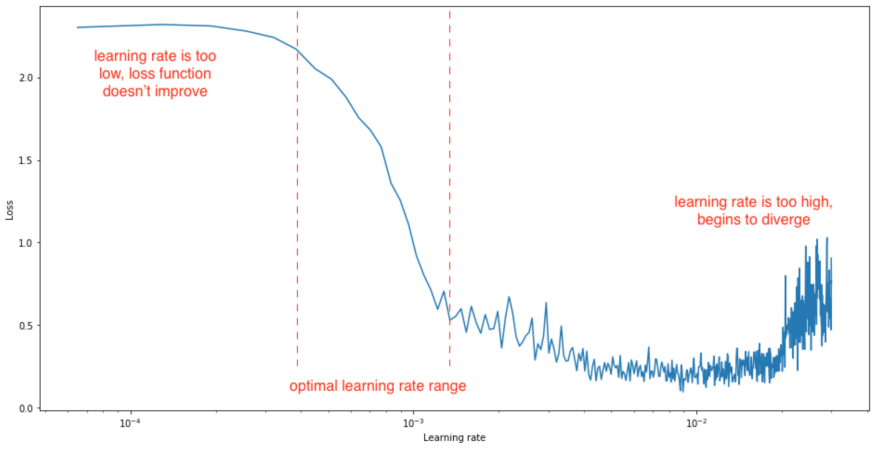
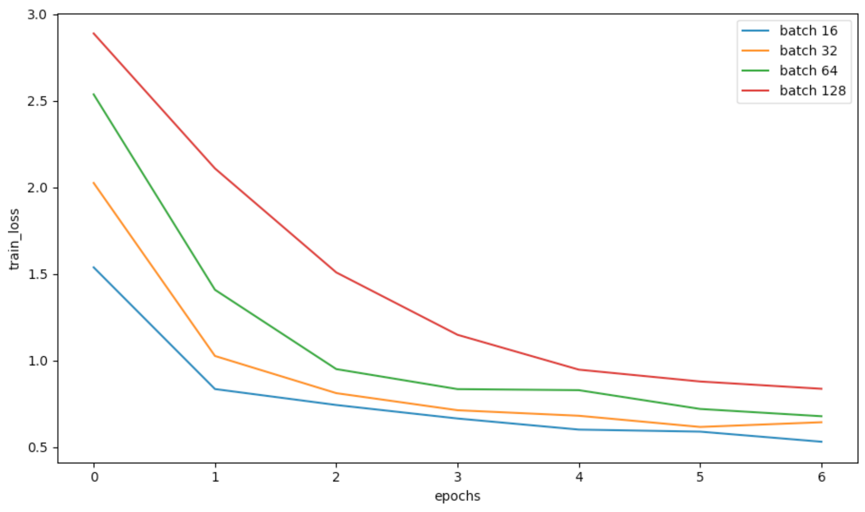


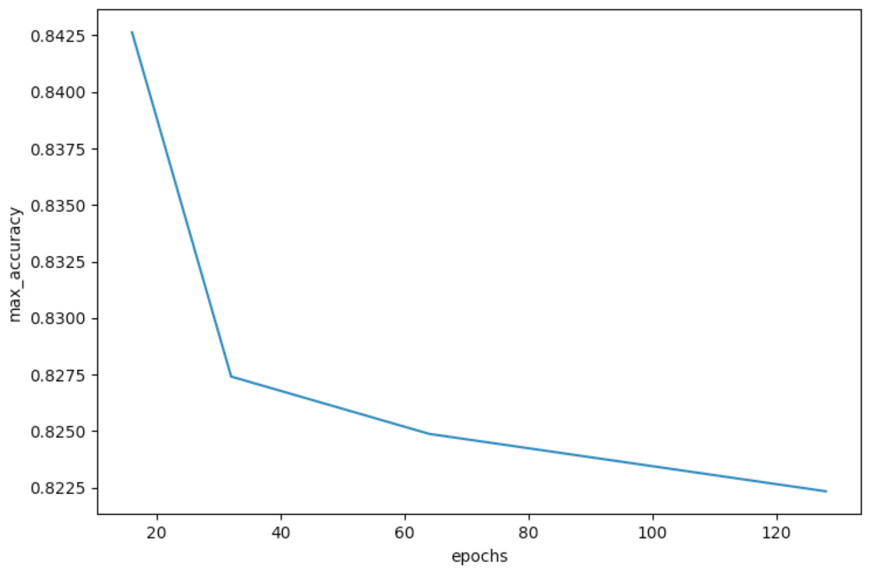
Figure 7: Image is taken from :[12]

6.4 Batch Size

For this experiment I set the parameters like following : epoch = 7, freeze =13 layer, lr = 0.001. And I changed the batch sizes from 16 to 128.Here is the loss behaviours of training phase.



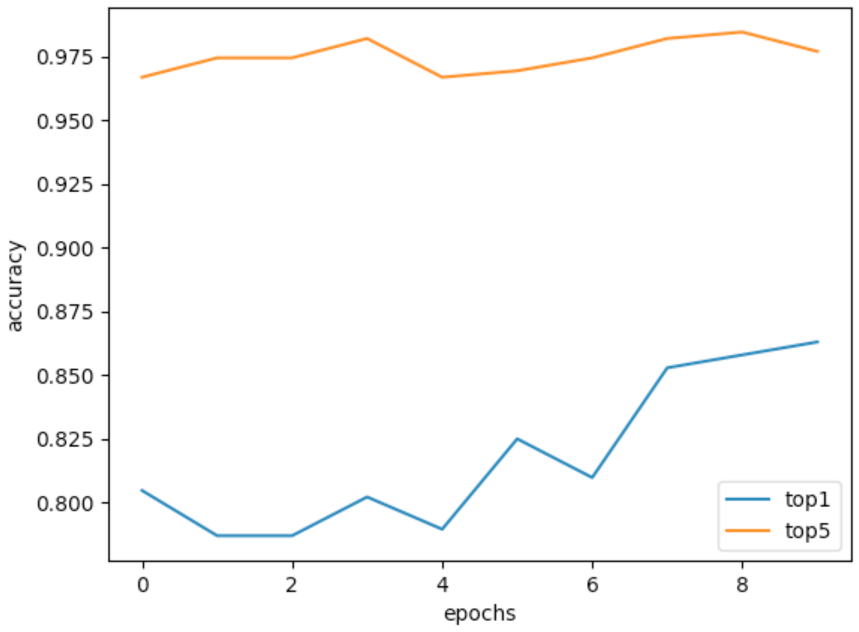
As you can clearly see increasing batch makes training more robust as it learns with bulks.In the same time learning in so small batches can cause the overfitting too.So here again, batch size also a parameter that should chose wisely.
After this experiment I wanted to see overall success of each batch size. And I plot the maximum test accuracy results for every batch training process.Results are like following:



As you can see increasing the batch slow down the process as it learns with big bulks, so it is less likely to reach the good results in few epochs. And also it is slower down the computational process too.

6.5 Top1 and Top5 Accuracy & Error

Top1 accuracy is more challenging job than top5. Top5 simply is a hint that our model is near to perfection so if it's good it means that "I am about to find keep going!".And also we consider this metric as even human accuracy is not %100 for top1.
And also most of the training datasets are labeling with humans, so we should consider this for our model too. Even labeling is not perfect so maybe machine is right and human is wrong. :)
So let's see their behaviour on plot in a good worked training phase. We will compare the test top1 and top5 scores.



As you can see from the beginning the seperation between the lines is more than the end of of the process. This simply means that model accuracy in improving in epochs. In first it near

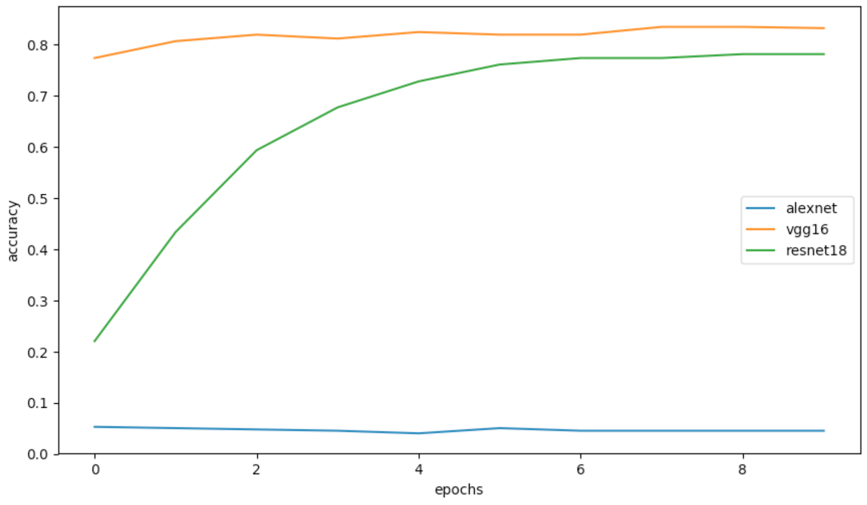
to find the right prediction at the end it slowly learns the predict better and this also means that weight evolving according to that.
So by considering top5 error we want to see that model is not that bad from the reverse side.We plot the top1 and top5 accuricies above in this section, as we already plot top1 and top 5 errors above in epoch section. Let's take a look at minimum top1 and top5 errors on those epochs here.

Epochs	Top1 Error	Top5 Error
5	0.191	0.024
10	0.150	0.010
30	0.119	0.009

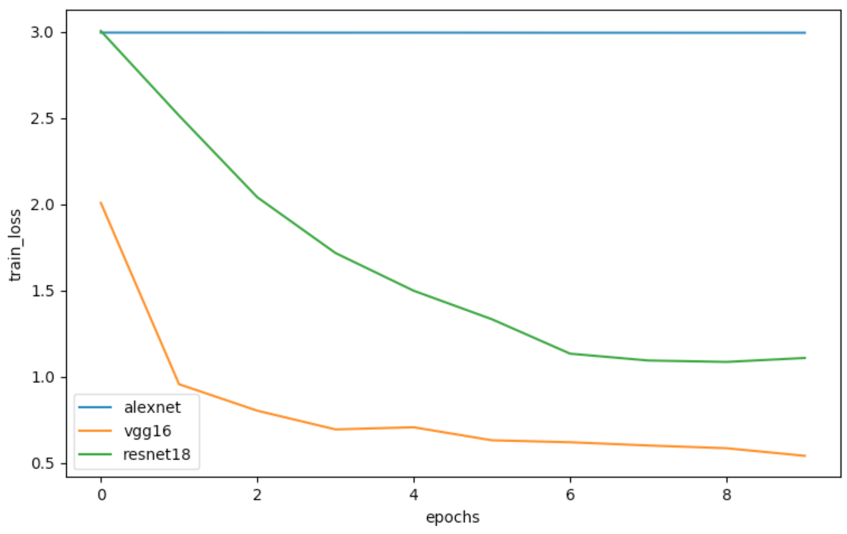
As you can see top5 error tends to reach better results in similar epochs than the top1 error.

7 Comparison With Other Models

I wanted to compare the VGG16 with trained model from the stracth and the fine tuned model in different architecture. For this sake I chose Resnet18 to fine tune and AlexNet for train. I keep epoch 10, batch size 32 and learning rate 0.001 for each model.As my optimum results are happened in 13 freezed layer in VGG16 I want to keep that proportion for resnet too. I freezed 16 layers of Resnet.



The image on above is the plot of accuracy comparison. As you clearly see most successfull model is vgg16.It is another good example to see that fine tuning work better than training from the stratch in less time. Let's consider their training phase behaviours too.For each model train phase loss plot is like following :



As you can see AlexNet is bearly learning in 10 epoch, it's not fixed but the changed numbers are so small that i can not seen in plot.On the other hand Vgg16 is far more better learner than resnet18, it looks like it will converge faster than resnet. And it's loss values are better than the others also.
After all, comparison conclusion is that VGG16 is a good base for fine tuning for this problem. Even tough it run slower than the others it learns better.

8 Further Extensions & Comments

To increase the accuracy as we talked above we can do data augmentation, also with more qualified computers and more time we can see the finish time by seeing the model where it converges.And after too many experiment we can be able to find the optimum model which is generalizes the data most.And also I would love to try this approach on the whole dataset that is given to us. I believe that it would be more accurate and good model after it really works on that too.
This was great experience for me to do such an assignment. I learned so many details about fine tuning and model training with this assignment I also had a chance to observe the results for different models and see the different behaviours of different models.
Other than that I observe the parameters effects on training phase, now I become more qualified the interpret the results by making research on the subject. And also I have seen that there is still so many things to learn about this subject. As I have so much interest on deep learning models, by the help of this assignment now I have more clear mind about how I will be more and more qualified about this stuff.
In addition to that I have seen that GPU has such a huge impact in training phase, I learned a lot about those hardwares too.
Thank you for this oppurtunity.

References

1. http://cs231n.github.io/convolutional-networks/
2. Simonyan, K. & Zisserman, A. (2014), 'Very Deep Convolutional Networks for Large-Scale Image Recognition', CoRR
3. https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035
4. He, K., Zhang, X., Ren, S.,& Sun, J. (2016). Deep Residual Learning for Image Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.
5. https://en.wikipedia.org/wiki/AlexNet
6. https://machinelearningmastery.com/transfer-learning-for-deep-learning/
7. Torrey Handbook, Chapter 11, Transfer Learning , Lisa Torrey and Jude Shavlik, University of Wisconsin, Madison WI, USA
8. https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algori
9. Perez, Luis and Jason Wang. "The Effectiveness of Data Augmentation in Image Classification using Deep Learning." CoRR abs/1712.04621 (2017): n. pag.
10. https://stats.stackexchange.com/questions/153531/what-is-batch-size-in-neural-network
11. https://discuss.pytorch.org/t/imagenet-example-accuracy-calculation/7840
12. https://www.jeremyjordan.me/nn-learning-rate/