Q5) Comparisons with Deep learning Models

a) Designed simple custom LeNet, and also used Resnet50:

LeNet model: We implemented a custom LeNet with 2 convolution layers and Max pooling with 3*3 filters and 2 fully connected layers (6 layers) while maintaining the 3 channels.

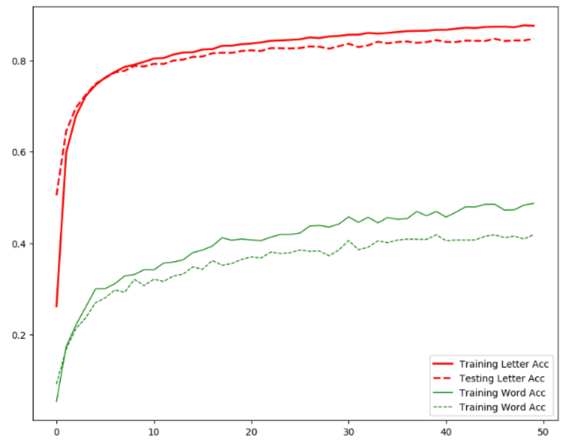
ResNet model: The main reason to use ResNet architecture is for the skip connections which help with vanishing gradient problem. Further information provided in the last section.

b) Plotting Letter wise and Word wise accuracies:

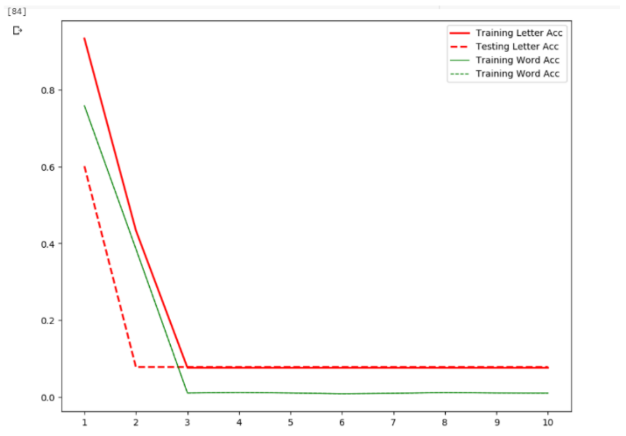
Plot of Letter wise and Word wise accuracies for ResNet and LeNet respectively:

LeNet Plots:

Adam



LBGFS



Training Letter Accuracy Peak: 87.67%

Testing Letter Accuracy Peak: 84.73%

Training Word Accuracy Peak: 48.7%

Testing Word Accuracy Peak: 41.8%

Training Letter Accuracy Peak: 96.78%

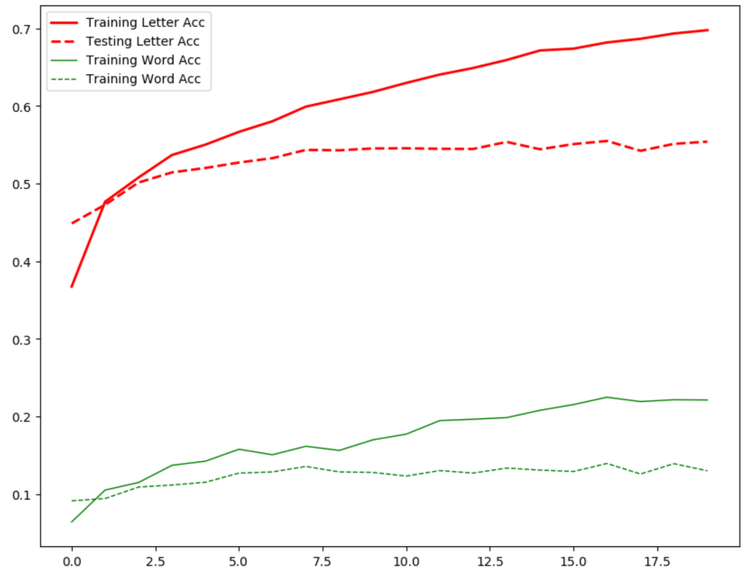
Testing Letter Accuracy Peak: 69.4%

Training Word Accuracy Peak: 78.9%

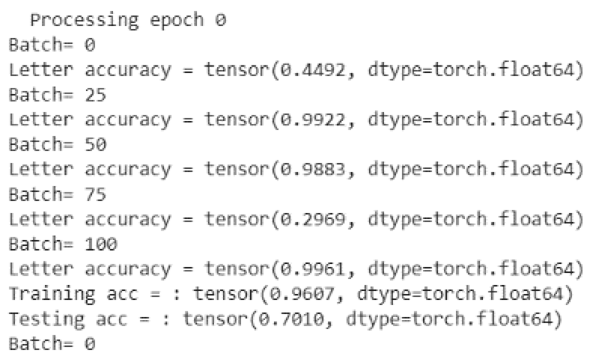
Testing Word Accuracy Peak: 37.4%

ResNet Plots:

Adam



Structure of results in notebook (LBFGS)



```
  Processing epoch 0
Batch= 0
Letter accuracy = tensor(0.4492, dtype=torch.float64)
Batch= 25
Letter accuracy = tensor(0.9922, dtype=torch.float64)
Batch= 50
Letter accuracy = tensor(0.9883, dtype=torch.float64)
Batch= 75
Letter accuracy = tensor(0.2969, dtype=torch.float64)
Batch= 100
Letter accuracy = tensor(0.9961, dtype=torch.float64)
Training acc = : tensor(0.9607, dtype=torch.float64)
Testing acc = : tensor(0.7010, dtype=torch.float64)
Batch= 0
```

Training Letter Accuracy Peak: 70.4%

Testing Letter Accuracy Peak: 55.01%

Training Word Accuracy Peak: 23.5%

Testing Word Accuracy Peak: 13.5%

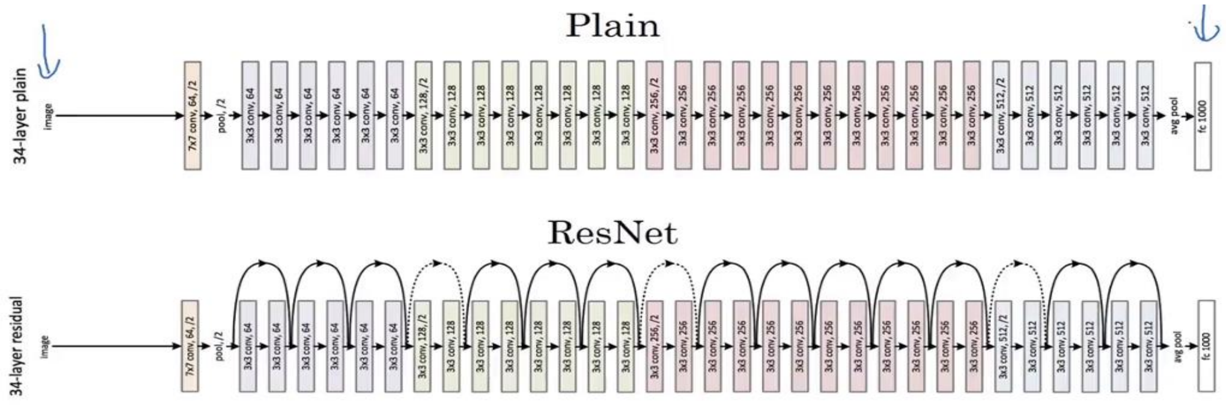
c) Comparing Letter wise and Word wise accuracies with ADAM and LBFGS:

Plots of Letter wise and Word wise accuracies for ResNet and LeNet respectively were shows above. ResNet plot of LBGFS was taking lot of time for each iteration. Adam finds better solution way faster and also converges faster. LBFGS solver is a true quasi-Newton method in that it estimates the curvature of the parameter space via an approximation of the Hessian. With larger input space, the second order calculations of LBFGS get pretty computationally heavy. That's the reason for the slow iteration speed.

d) Why ResNet and LeNet? Design and Implementation of models ResNet and LeNet:

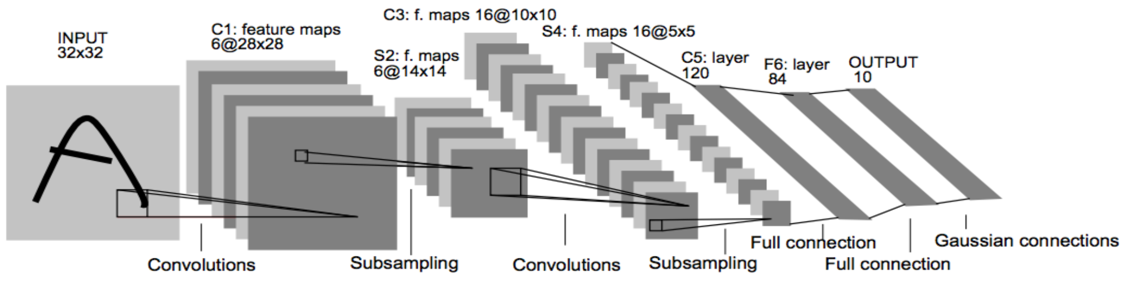
ResNet:

We chose ResNet mainly to understand the skip connections and how it deals with Vanishing gradient. With networks going deeper, the issue of vanishing gradient becomes much more crucial. Therefore it is important to dig into the issues and how models like ResNet solve those issues.



Compared to normal plain network, ResNet uses skip connections. They help is two things:
 Mitigating Vanishing Gradient
Helps model learn identity function which ensures that higher layer will perform at least as good as the lower layer, and not worse.
The ResNet-50 model consists of 5 stages each with a convolution and Identity block. Each convolution block has 3 convolution layers and each identity block also has 3 convolution layers. The ResNet-50 has over 23 million trainable parameters. Drop used in ResNet- 50 is 0.7x, 70% drop chance.

LeNet: We used a simple LeNet-5 which has 2 convolutions and 2 maxpool layers. Followed by 3 fullyconnected layers. We maintain same number of channels and pad it accordingly. Even though LeNet is simplest deep learning model out there, it does a great job for dataset with small features like out 16*8 input.


INPUT 32x32
C1: feature maps 6@28x28
S2: f. maps 6@14x14
C3: f. maps 16@10x10
S4: f. maps 16@5x5
C5: layer 120
F6: layer 84
OUTPUT 10
Convolutions
Subsampling
Convolutions
Subsampling
Full connection
Full connection
Gaussian connections