**Final Project Report**

A**nswer in your Report:** Why should we add the momentum to SGD? What is the main two improvements Adam has comparing to the simple SGD? What was the best optimization algorithm and best learning rate you've found?

**Answer:** SGD can cause zig-zagging dynamics in the gradient updates, that is because we estimate the derivates on a small batch. One way to tackle this problem is to add momentum, which is an accumulating parameter of past updates, analog to momentum in physics, where acceleration can be accumulated from past movements. As a result, momentum accelerates converging time, and can find a better local minimum.

Adam, which utilizes momentum among other features such as bias-correction and RSMProp, is straightforward to implement, has faster running time, low memory requirements, and requires less tuning than any other optimization algorithm.

The optimizer that gave us the best results is Adam, and the best learning rate we found is 8e-4.

**Answer in your Report:** What is the purpose of batch normalization (why do we use it) Did the Batch normalization improve the network performance?

**Answer:** Batch Normalization is the process of transforming each Batch mean to zero, and standard deviation to one. Hence bringing the data to common scale without distorting its shape. we use it for various reasons:

* It leads to faster and better convergence
* Serves as a regularization method.
* Decrease the importance of weight initialization.
* Allows for higher learning rate- In the previous experiment, a learning rate value of 1e-3 caused the networked to become untrainable. however, after adding batch normalization, we were able to fine tune the LR further, leading to significantly better results.

(Our network will use 1e-3 from now), also we were able to raise our filters per layer to 128 as a result.

**Answer in your Report:** Why should we use regularization? How does the regularization affect the train accuracy and loss? How does it affect the test accuracy and loss? What was the best regularization method?

**Answer:** Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model’s performance on the unseen data as well, meaning improving the test accuracy and decreasing test loss, we tried various regularization methods:

**Dropout-** We tried this regularization method with various P values for hidden and input layers. It had negative impact on the performance of the network for all configurations we tried. Not only the accuracy was degraded but also the convergent rate was significantly slower

**L1, L2-** The best performance was achieved with L2 value of 1e-3, the network was able to converge faster and with better accuracy

**Answer in your Report:** Did the data augmentation improve the model performance? Mention which augmentation you used and why.

**Answer:** We tried Crop, but the performance of the network was degraded, eventually the best results we achieved were with image normalization.

**Answer in your Report:** What is your best architecture? what its best accuracy and loss on the test set? Compare your result to assignment 2. How did you manage to improve the model?

**Best architecture:** Filters\_per\_layer= [128], layers\_per\_block = 4, pool\_every = 2, hidden\_dims= [200,200]

**Best LR:** 1e-3

**Regularization used:** L2

**Augmentation used:** Image Normalization

**Additional Methods used:** Batch Normalization

**Methods unused:** Dropout, L1

**Best Accuracy:** 0.78

**Best Loss:** 0.68

Compared to Assignment 2, we managed to improve accuracy by 10%.

we increased filter size, pooled every 2 layers, and increased hidden dims to [200,200]. In addition, batch normalization, L2, and Augmentations were used to improve network performance.

**Summarize this Blog in the report**-

be sure you answer these questions:

1. What is the ‘Res’ for in ResNet?
2. What is the main innovative idea presented in ResNet?
3. Which problem this unique architecture trying to solve?

**Answer:**

ResNet, which stands for Residual-Network, is one of the most groundbreaking deep-learning architectures of the recent years.

As CNN networks gotten deeper and deeper, the vanishing gradient problem became more prominent- as the gradient is backpropagated to earlier layers, the gradient gets smaller because of the repeated multiplications, heavily degrading networks performance.

The main innovative idea presented in ResNet is the approach to solve the vanishing gradient problem. The core idea is to add “identity shortcut connection” that skips at least one layer.

This idea was in fact introduced in an older network called Highway Network, where these shortcuts are parameterized to control the flow of information in them. However, because Highway Network did not perform any better than ResNet, it seems that its more important not to block these connections than allowing flexibility. Following this intuition, this concept of residual blocks was refined to a pre-activation variant, in which the gradients can flow through the shortcut connections to any other earlier layer.

**Answer in your Report:** What was ResNet50 accuracy and loss? Is it overfit/underfit/well-fit the data?

**Answer:** With ResNet50 with no pretraining, we managed to achieve test accuracy of 0.76, And loss of 0.74, Which is slightly below our own model performance, it seems to have well fitted the data.

**Answer in your Report:** What was the pretrained ResNet50 accuracy and loss? Is it overfit/underfit/well-fit the data? Has it got better accuracy than the non-pretrained ResNet?

**Answer:** With pre-trained ResNet-50, we managed to achieve test accuracy of 0.78 and loss of 0.66, which is slight above our own model performance and resnet50 without pretraining.

It appears that the pretrained model achieves better accuracy right off the start with the first epoch (0.68 test accuracy in the first epoch) and converged faster to its peak.

However, the difference in performance wasn’t significant, and the pre-trained and non-pre-trained model achieved similar results.

**Other models we tried:**

**Densenet**- Similar Architecture to ResNet but connects each layer to every other layer. Achieved 0.79 test accuracy and 0.59 loss, it was able to converge quickly to its peak performance (after 11 epochs), this model took the longest to train

**VGG**- Classic CNN architecture, based on Alexnet. Pioneer of large quantity of smaller kernels approach. Achieved 0.81 test accuracy, 0.67 test loss. It is the best performing model in all our experiments

**Answer in your Report**: Write a short summarization of your attempts. What worked for you? which architecture and hyperparameters led to the best performance?

**Answer:**

* We started with filter size of 32, 4 layers per block, pool every 4, as per assignment 2.
* We added Adam optimizer, which improved our results.
* We used batch normalization, allowing us to increase learning rate and filter size, which in turn improved our results by 10%, which was the most significant improvement we achieved.
* The only augmentation we used is image normalization which slightly improved performance, other types of augmentations didn’t work for us
* We used L2 for regularization, we tried dropout but it degraded network performance
* Resnet-50 with no pre-training achieved slightly worse performance than our model
* Resnet-50 with pre-training achieved slightly better performance than our model
* We also tried densenet, it achieved slightly better performance than our model (2% increase in accuracy)
* We also tried VGG, which achieved 81% accuracy, the best result we got in all of our experiments.