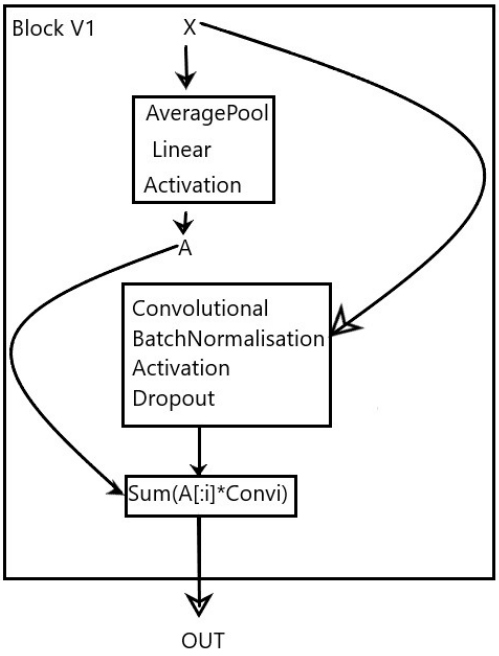
Model:

Backbone

My backbone consists of 2 types of blocks.

Block V1

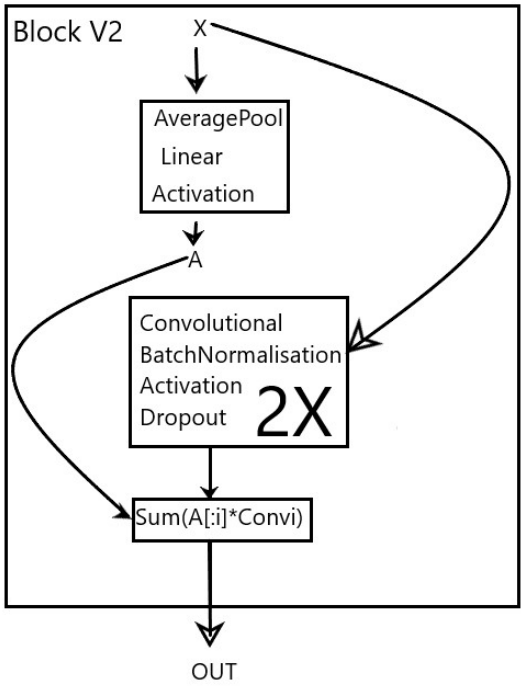


A is computed by average pooling X then applying linear function to the output and activate the result with ReLu.

Each conv block contains convolutional, layer batch norm, activation, and dropout.

The output of a and convolutions is then multiplied by each other and summed.

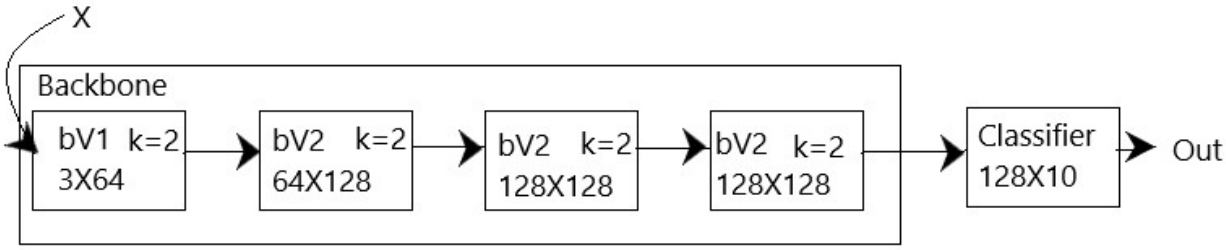
Block V2



It's pretty much the same block but each convolutional block goes through the convolutional cycle twice.

Afterwards the output value again average pooled and putted in classifier block which is in my case is basic linear function.

My final Architecture:



It consists of 1 block of version1 and 3 version2 blocks and then the output from them is sent to the classifier.

With these parameters I used for testing:

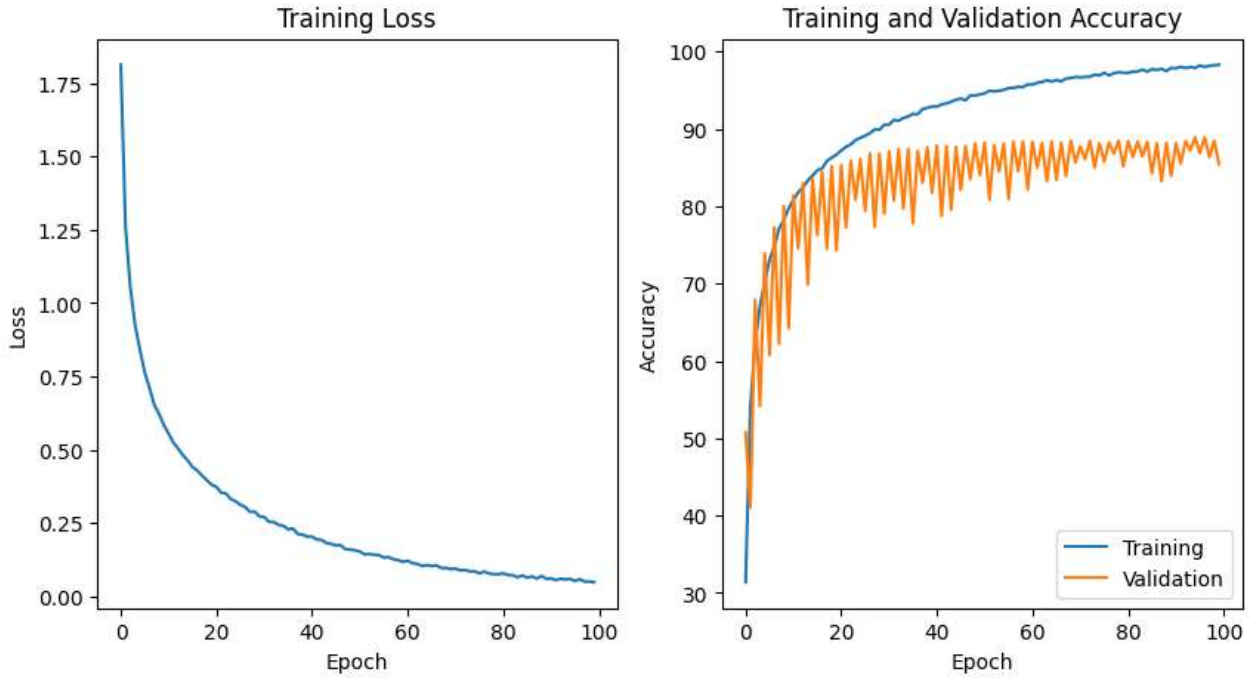
Batch_size=100

```
criterion = nn.CrossEntropyLoss()
num_epochs = 100

# Initialise the model
model = CustomBlockCNN().to(device)

# Higher initial learning rate
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9, weight_decay=1e-7)

scheduler = optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=num_epochs, eta_min=0)
```

I was able to achieve highest Validation Accuracy of 88.92%

```
Epoch 97, Loss: 0.0587, Train Accuracy: 98.02%, Validation Accuracy: 88.92%
```

And here are the graphs for the Loss and train and validation accuracy.



By looking at the curve of validation we can clearly see that the model is a bit overfitted and could be improved, which might be achieved by decreasing learning rate or increasing the number of k's.