All edits to BaseNet which lead to a significant accuracy improvement must be listed in the report. You must include at least one ablation study for such an edit (or a combination of edits), that is, submitting results with and without the edit to Kaggle, and reporting the performance improvement.

Below edits all are trained with the following settings:

EPOCHS = 20 # small since many edits tested

TEST\_BS = 256

TRAIN\_BS = 32

**BaseNet w/o any changes =**

Training loss & validation accuracy for final 3 epochs of network training:

[18] loss: 2.785

Accuracy of the network on the val images: 27 %

[19] loss: 2.769

Accuracy of the network on the val images: 26 %

[20] loss: 2.745

Accuracy of the network on the val images: 26 %

**Change 1 =**

Data normalization and Data augmentation added for training data (as shown below):

* transforms.Normalize(train\_mean, train\_std)
* transforms.RandomHorizontalFlip(0.5)
* transforms.RandomCrop([32,32])

Training loss & validation accuracy for final 3 epochs of network training:

[18] loss: 2.813

Accuracy of the network on the val images: 28 %

[19] loss: 2.798

Accuracy of the network on the val images: 27 %

[20] loss: 2.790

Accuracy of the network on the val images: 27 %

Improvements: Very small accuracy improvement - essentially is the same acc./loss, however, likely helped when running with more epochs + when network has more layers, etc.

**Change 2 =**

Conv output filters changed from 6 & 16 🡪 64 & 128

+ BatchNorm2d() layer added before each ReLU() layer

+ learning rate changed from 0.005 🡪 0.01, and momentum changed from True 🡪 False

Training loss & validation accuracy for final 3 epochs of network training:

[18] loss: 1.647

Accuracy of the network on the val images: 41 %

[19] loss: 1.600

Accuracy of the network on the val images: 41 %

[20] loss: 1.558

Accuracy of the network on the val images: 30 %

Improvements: From around 25% (training accuracy) to 30-40% percent – the accuracy gets up to 42%, but drop down to 30% on the last epoch, likely due to overfitting

Running it again we see:

[18] loss: 1.365

Accuracy of the network on the val images: 38 %

[19] loss: 1.358

Accuracy of the network on the val images: 39 %

[20] loss: 1.344

Accuracy of the network on the val images: 39 %

Hard to predict, without using more epochs – however, we definitely have a strong improvement here.

**Change 3 =**

Added 2 more conv-sequentials with the same format as the format as the current 2; Conv2d(), BatchNorm2d, then ReLU. Thus, the conv output filters were changed from 64 & 128 🡪 64 & 64 & 128 & 256

+ avg\_pool2d() layer added after the conv-sequentials (with pool size = 4)

Training loss & validation accuracy for final 3 epochs of network training:

[18] loss: 1.381

Accuracy of the network on the val images: 46 %

[19] loss: 1.342

Accuracy of the network on the val images: 49 %

[20] loss: 1.304

Accuracy of the network on the val images: 51 %

Improvements: From around 40% (training accuracy) to around 50% percent.

**Change 4 =**

Changed the 2nd, 3rd, and 4th conv-sequences into res blocks

Training loss & validation accuracy for final 3 epochs of network training:

[18] loss: 0.922

Accuracy of the network on the val images: 53 %

[19] loss: 0.870

Accuracy of the network on the val images: 52 %

[20] loss: 0.819

Accuracy of the network on the val images: 54 %

Improvements: Does similar (51% to 54% in this example), however, improved on overfitting (seen when ran with more epochs).

**Ablation Study =**

For (**Change 3**), we ran out model on the test data, and uploaded our predicted labels to Kaggle, this is the following output (before the change vs. after):



