HW 2 Deep Neural Networks, Classifiers & Features

[redacted]

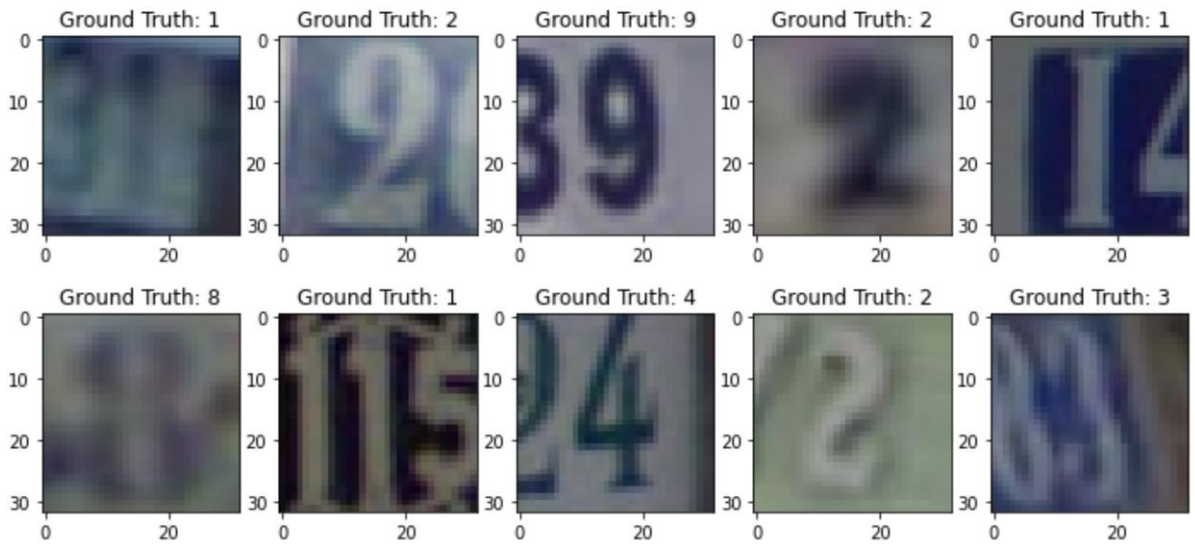
[redacted]

1 Part 1: Design and Build a CNN Classifier

Done with *colab*. Open code file there to read with ease (markdown syntax is different).

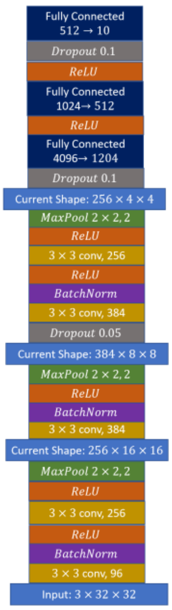
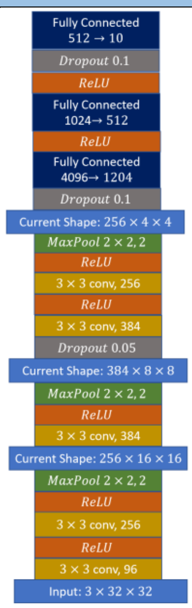
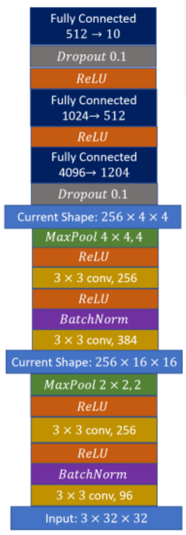
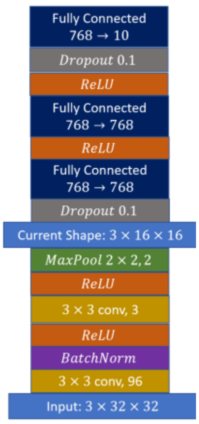
1.1 Question 1

We were to load the SVHN data set and display 5 images. We chose to show a little more:



1.2 Question 2

We were to build a CNN which will classify the digits from the images.
We experimented with a few architectures all based on Alex-net.

<table>
<tr><th>AlexNet Modified</th><th>AlexNet Modified NoBatchNorm</th><th>AlexNet Modified Smaller</th><th>Mostfly FC</th></tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</table>

Calculating the Number of parameters per layer:

- Convolution layer with kernel $K \times K$ with N input channels and M output channels:

$$\underline{K^2 \cdot N \cdot M + M}$$

- Batch Normalization layer with N input channels:

$$\underline{2 \cdot N}$$

- Fully connected layer with input of size N and output of size M:

$$\underline{N \cdot M + M}$$

For *AlexNet Modified we reached* 34,720,896 parameters (without taking batch normalization into account).

1.3 Question 3

We were asked to train the model and assess the results.
We first trained 4 CNN models with similar hyper-parameters:

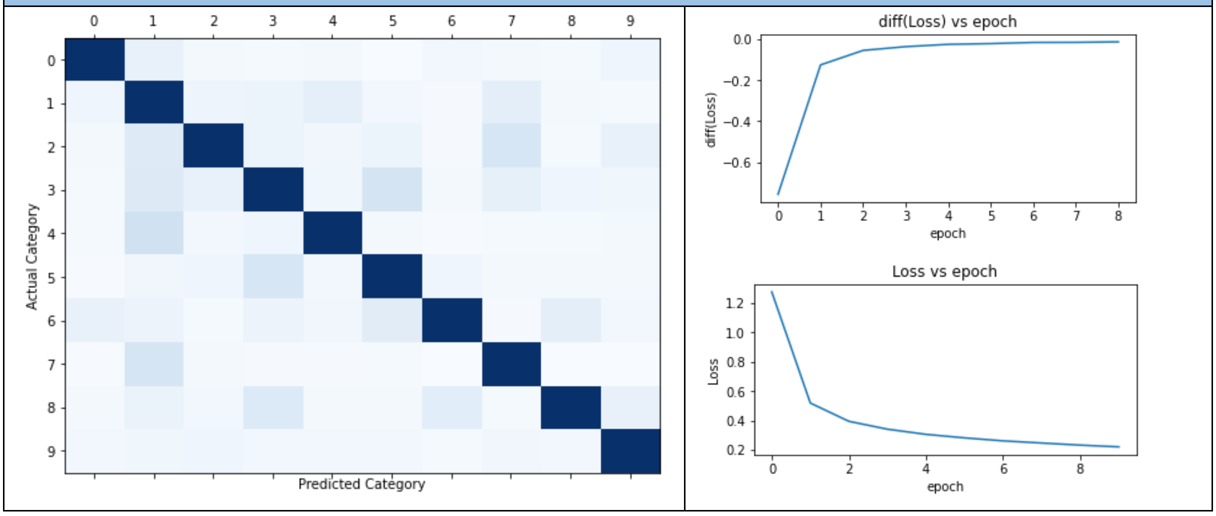
- Batch size – 128
- Learning Rate - $1e - 4$
- Epochs – 10

The results:

	AlexNet Modified	*AlexNet Modified NoBatchNorm*	*AlexNet Modified Smaller*	*Mostfly FC*
Accuracy [%]	94.574	92.354	93.850	79.143
~Average Epoch Time [s]	86	85	80	65

We also plotted the confusion matrix and *loss vs epoch* graph for each model to make sure that 10 epochs were enough for convergence:

Learning Graphs for *AlexNet Modified*



It is not surprising that the deepest model provides us with the best accuracy, but we removed a whole convolution block between *AlexNet Modified* and *AlexNet Modified Smaller* without having much change in accuracy and average epoch time. It seems the middle convolution block in *AlexNet Modified* is not very active.

We then changed the hyper parameters and retrained *AlexNet Modified* to view their effects

	AlexNet Modified		
Batch Size	512	1024	1024
Learning Rate	1e-4	1e-3	5e-4
Epochs	5	5	10
Accuracy [%]	88.372	76.375	92.272
~Average Epoch Time [s]	79	78	79

Unsurprisingly, increasing batch size reduced the average epoch time. The batch size technique was initially created to avoid long computation times. The negative effect of using batches is that the gradient is not as “true”.

Also, we noticed that increasing the learning rate above a certain threshold had significant an effect on the accuracy.

2 Part 2: Analyzing a Pre-trained CNN

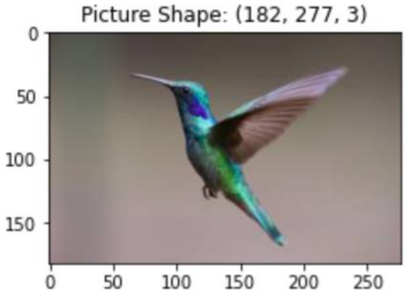
2.1 Question 1

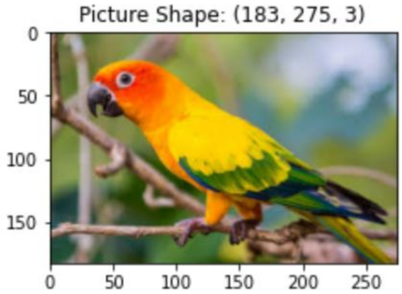
We loaded the VGG16model from torch via the following commands:

```
device = torch.device("cuda:0" if torch.cuda.is_available() else
"cpu")
VGG16model=tv.models.vgg16(pretrained=True, progress=True).to(device)
VGG16model.eval();
```

2.2 Question 2

We were given two bird images and asked to display them.





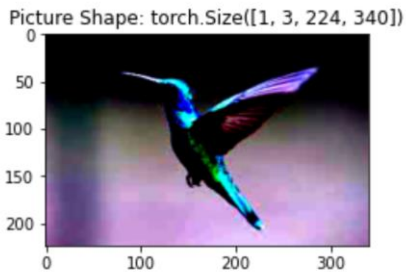
2.3 Question 3

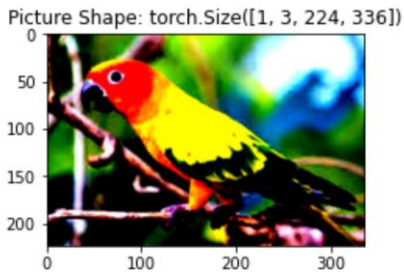
We were asked to preprocess the bird images to fit VGG16's architecture.
Pictures in VGG16 need to be at least 224x224 and sorted to mini-batches of $(3 \times H \times W)$.
Also, they need to be in range of [0,1], normalized using

$$\text{mean} = [0.485, 0.456, 0.406], \text{std} = [0.229, 0.224, 0.225]$$

And un-squeezed to have an additional dimension.

```
def fit2VGG16(images,device):
    preprocess = transforms.Compose([
        transforms.Resize((224)),
        transforms.ToTensor(),   #also converts image to 0-1 instead
of 0-255
        transforms.Normalize(mean=[0.485, 0.456, 0.406],std=[0.229,
0.224, 0.225])
        ])
```





2.4 Question 4

We fed the images (forward pass) to the model and checked the outputs using class string labels from test files we downloaded (attached with HW).

The network has detected the hummingbird successfully but failed with the lovebird parrot. We are no bird experts, but a lorikeet looks somewhat different:

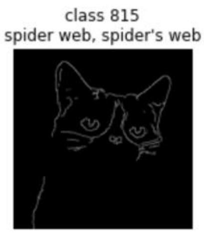
2.5 Question 5

We were asked to load an image from the internet, pass it through VGG16 and show the output. We chose *'Grumpy Cat'*.

'Grumpy Cat' is actually $\frac{1}{2}$ 'Tabby' and $\frac{1}{2}$ 'Calico' so VGG16 had gotten him wrong, but close enough we guess.

2.6 Question 6+7

We applied transformations to ‘Grumpy Cat’ to create new images and passed them through VGG16 to see the transformation’s effects on the classifier.

Original	Gaussian Blur	Canny	Rotate 45	Rotate 180
				

Original	Enhance Green	2 LAB colormap	Threshold [50,255]	Threshold [127,255]
				
				

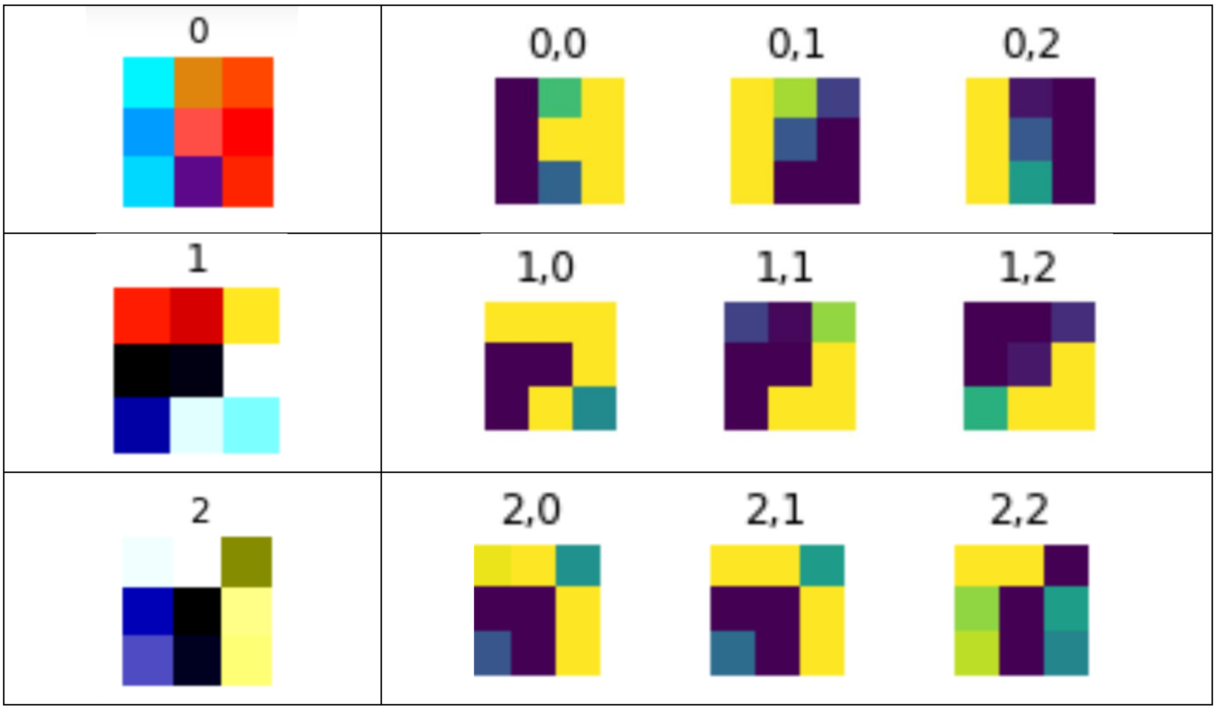
All in all the VGG16 did a great job.
It would seem VGG16 works rather well with affine transformations, but it heavily relies on the color scheme of the object.

We can also presume that all the admiral pictures fed into VGG16 while training were of white people with black hats.

2.7 Question 8

We were asked to show the first three filters of the first layer in VGG16 and plot their response.

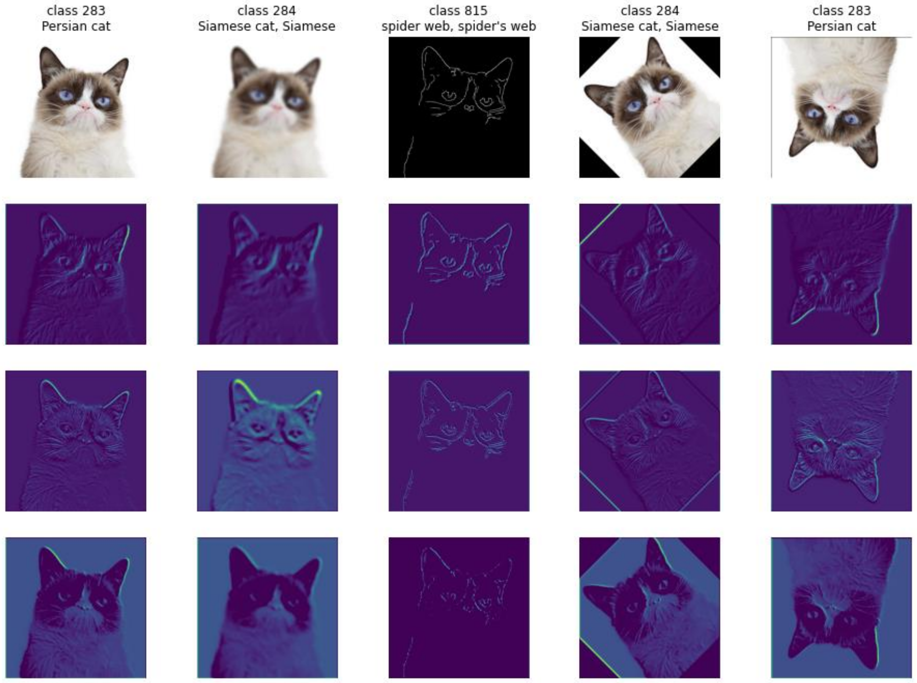
Filter in RGB colorspace	R G B channels of filter

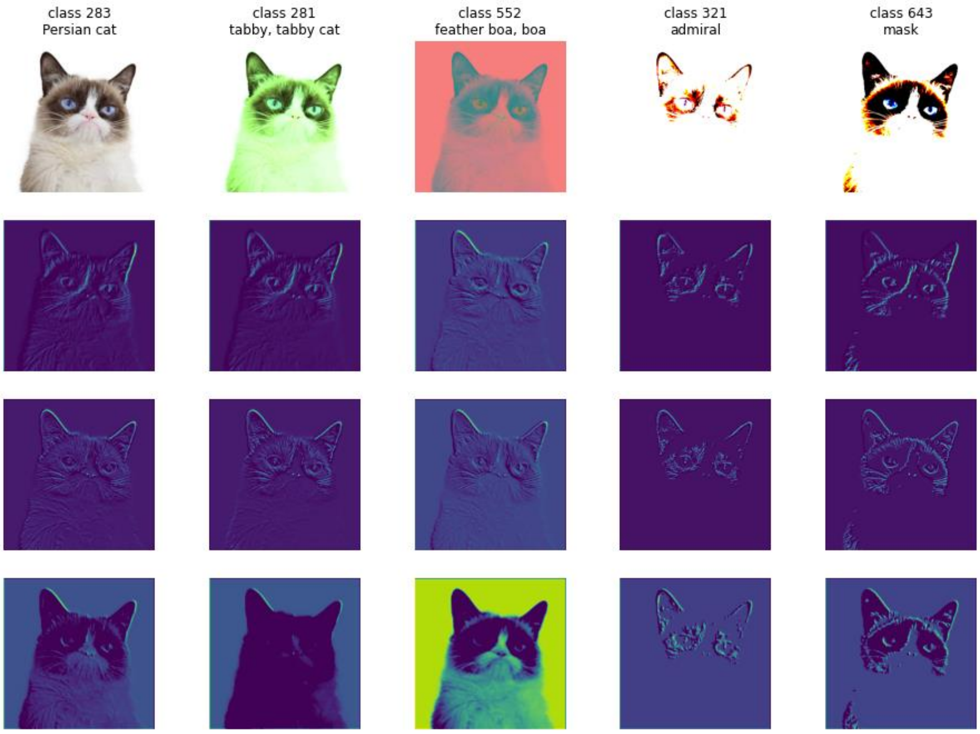


It would the first three filters are *'edge finders'*.
The first filter clearly finds derivatives along the *x* axis.
The second and third filters are prone to activate edges along a $\pm 45°$ direction from the *x* axis.

We do not know why it is so, but the derivatives directions vary across the channels in each filter.

Below we had plot cumulative response to the filters we showed. The response certainly strengthens our claim as to what the filters do.
The third filter seems to enhance the significance of some colors over the others.





2.8 Question 9

We were to load dogs and cats pictures provided with the assignment and pass them through VGG16 in a way where we could extract feature vectors from an FC layer within.

We chose the last layer of the FC part in VGG16 because the classifier was trained to classify different types of cats and dogs.

Hence for each image passed through VGG16 we obtained a feature space of 1×1000.

The pictures are presented below:

2.9 Question 10

Finally, we were asked to create a classifier for cats and dogs based on the features we extracted in $Question$ 9 using SVM technique (sklearn.svm.LinearSVC), and test the classifier on pictures from the web.

Our classifier seems to work well on real cats and dogs but the cat with the hat was misdiagnosed. May have something to do with the black nose.