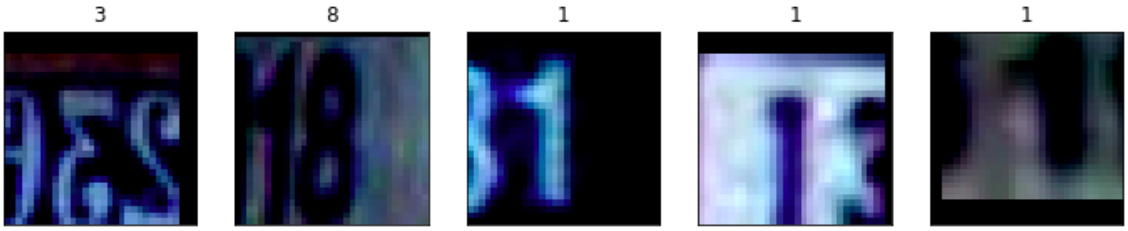
Computer vision
Homework 2

Ido Notuv 305242968

Sarah Buzaglo 931161798

Part 1 - Design and build a CNN classifier

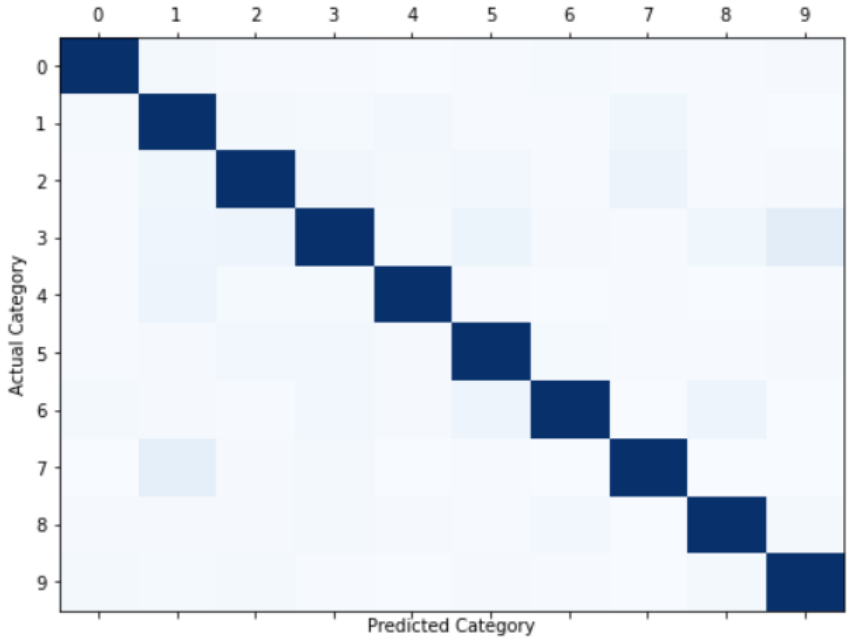
1) Display 5 images from the train set.



2) Tutorial training

What is the accuracy on the test set?

```
test accuracy: 94.614%
```



What classes are most confusing for this model?
We can observe:

- **a confusion between 3 and 9**
- **a confusion between 1 and 7**
- **a confusion between 2 and 7**
- a slight confusion between **8 and 6**
- a slight confusion between **3 and 5**

3) Design a Convolutional Neural Network (CNN) to classify digits from the images.

Our model has 1167882 trainable parameters.

Model architecture:

```
BasicCNN(
  (conv_layer): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (6): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): ReLU(inplace=True)
    (11): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (12): Dropout2d(p=0.05, inplace=False)
    (13): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (14): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (15): ReLU(inplace=True)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): ReLU(inplace=True)
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layer): Sequential(
    (0): Dropout(p=0.1, inplace=False)
    (1): Linear(in_features=4096, out_features=10, bias=True)
  )
)
```

We chose Relu activation function.

All the filters spatial dimensions are of spatial size $3x3$.

We used one fully connected layer size $(4096x10)$, after the imaged passed in the feature extractor layer we got a $4x4x256$, after flattening this is a vector size of $1x4096$ we have 10 classification categories representing 10 digits hence the fc output size.

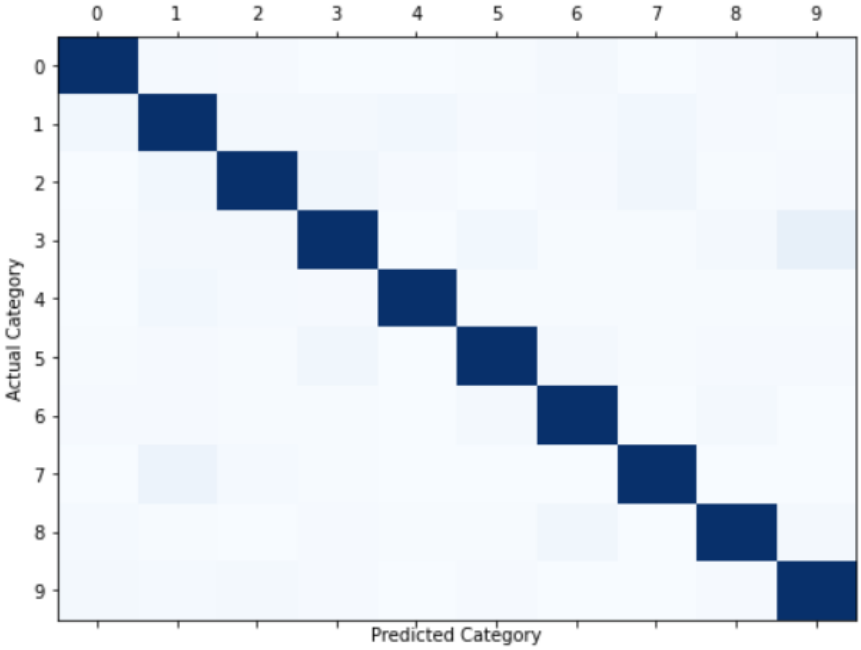
On the basis of the tutorial CNN, we also add a few improvements:

- Normalize with SVHN mean [0.44154901 0.44605679 0.47180032] and std [0.12072468 0.12448521 0.10762194]
- Increase to 0.1 and add of dropout layer which helps prevent overfitting.
- Learning rate optimal: 4e-4
- Batch size optimal: 64
- Use of a validation set

- Data augmentation: use of random rotation with an angle of 10.
- We used less layers on the fc part of the layer to prevent risk of overfitting.
- We used a learning rate scheduler to reduce the learning rate as long as the learning advance, the idea is to start with a big learning rate to get close to the area of the global minimum and then perform fine tuning on the weights.

4) Train the classifier

```
test accuracy: 95.863%
```



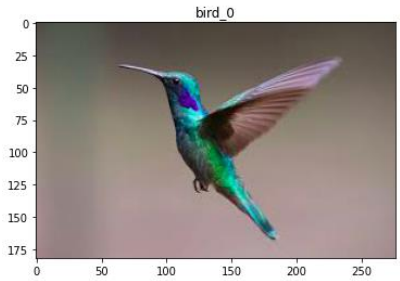
Part 2 - Analyzing a Pre-trained CNN

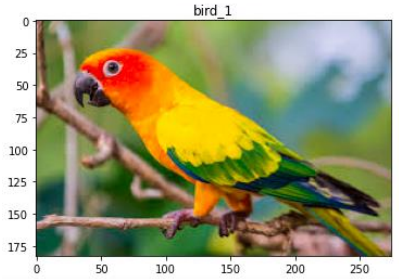
1) Code. Load a pre-trained VGG16
2) Code. Load the images
3) Pre-process the images to fit VGG16's architecture:

- Normalization (mean = [0.485, 0.456, 0.406], std = [0.299, 0,224, 0,225])
- Type to tensor
- Resize to 224x224

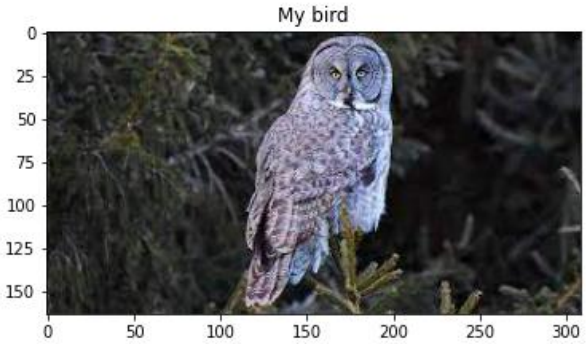
4) We get that output:

```
im1 =  ('94', 'hummingbird')
im2 =  ('90', 'lorikeet')
```





5) A new bird

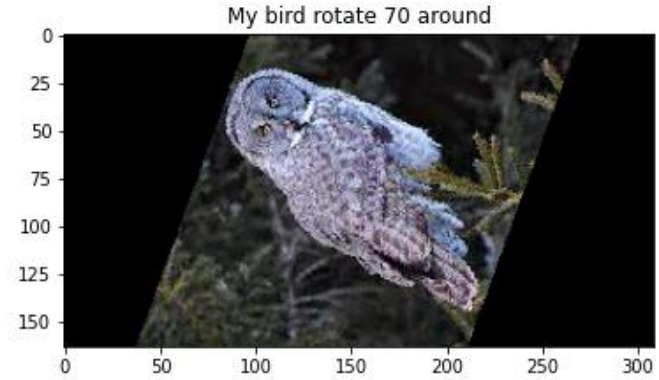



```
my bird =  ('24', 'great_grey_owl')
```

Comment: true result, this is a grey owl

6) + 7) Transformations

One geometric transformation: 70 degre rotation




```
my_bird_70deg =  ('43', 'frilled_lizard')
```

Comment: wrong result, this is not a frilled lizard

One color transformation: contrast (factor 1.7)


```
my_bird_contrasted =  ('24', 'great_grey_owl')
```

Comment: true result.

One filter: the blur filter (GaussianBlur=2)


```
my_bird_blur =  ('210', 'German_short-haired_pointer')
```

Comment: wrong result, this is not a German short haired pointer.

Conclusion:

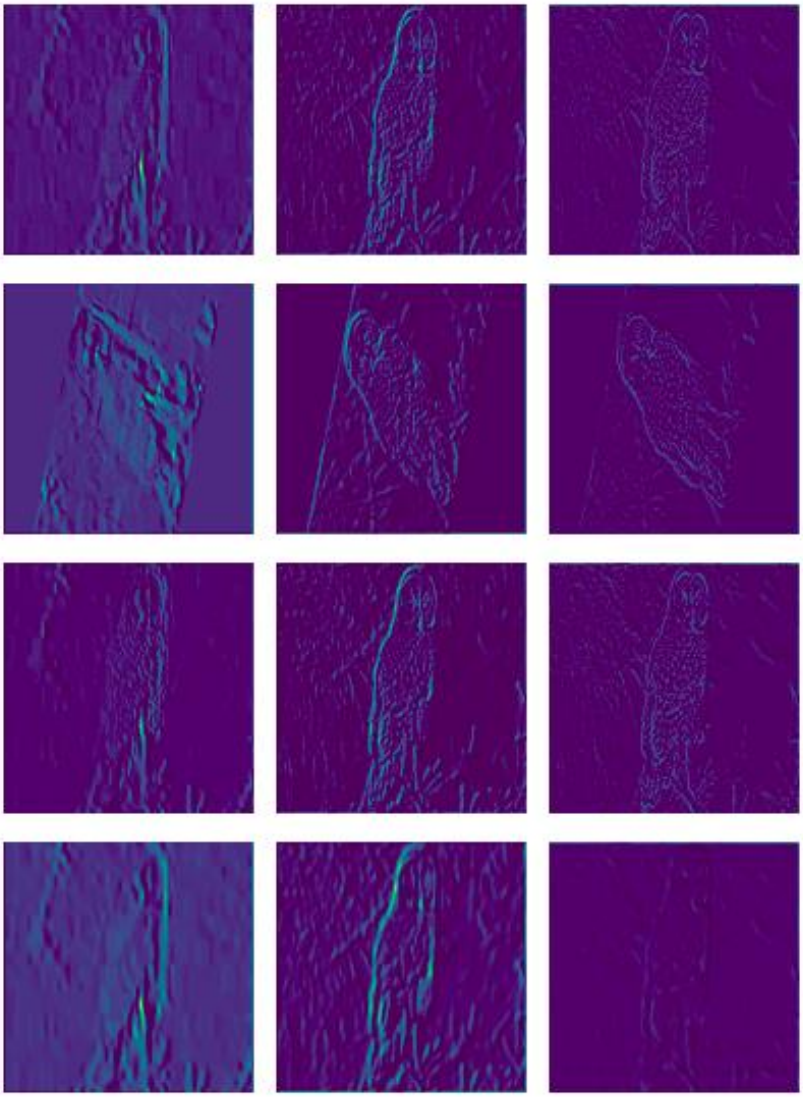
From the first transformation, we can conclude that the model was trained mainly on a horizontal view of owls, this follows our intuition on how owls were photographed, in there

habitat on the trees and upright, hence we can come to the conclusion that in the dataset the model was trained on there were very few near vertical owl photos.
From the second transform we can see that the model is robust to illumination changes, we can conclude that in the dataset the model trained on various illumination conditions.
From the third transform we conclude that the blurring deformed the texture and the characteristics of the owl that made the model predict the image as a German Short Haired Pointer.

8) Filters

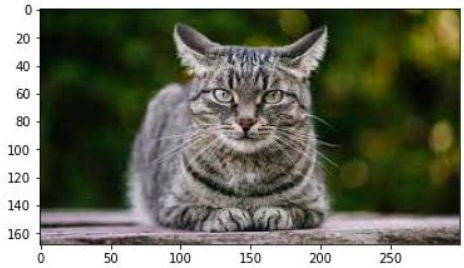
The first 3 filters in the first layer of VGG16:

We can see in highlighted areas what the filter detects at each stage as we learned in tutorial 4.

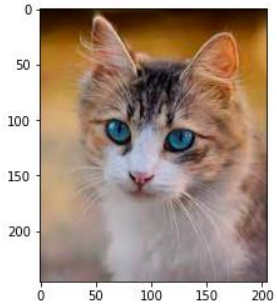


We suspect that filter 3 emphasizes the owl feather texture, this is further supported by the weak filter response to the blurred image and also not affected by illumination.
We suspect that filter 1 responds to edges in the rotated image, hence the filter emphasizes edges in a certain direction.
The filters responses for the contrasted image is similar to the filters responses for the original image, even a little stronger. Augment the contrast allow us to have stronger filter responses.

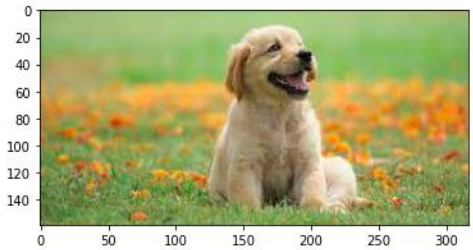
9+10) We use 2 ways: k-nearest neighbor (with k=3) and linearSVM and we get correct results



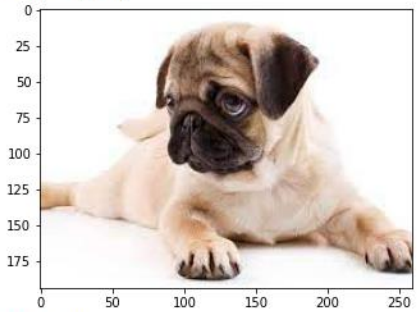
```
clf = [1.]
neigh = [1.]
```



```
clf = [1.]
neigh = [1.]
```



```
clf = [0.]
neigh = [0.]
```



```
clf = [0.]
neigh = [0.]
```