# DSSC - CVPR

Grimaldi Marco Alberto

February 2020

## Introduction

This project requires the implementation of an image classifier based on convolutional neural networks. It's dived in three parts which I'll cover one at time. The data-set is given and has been already split into training and test sets. I used MATLAB.

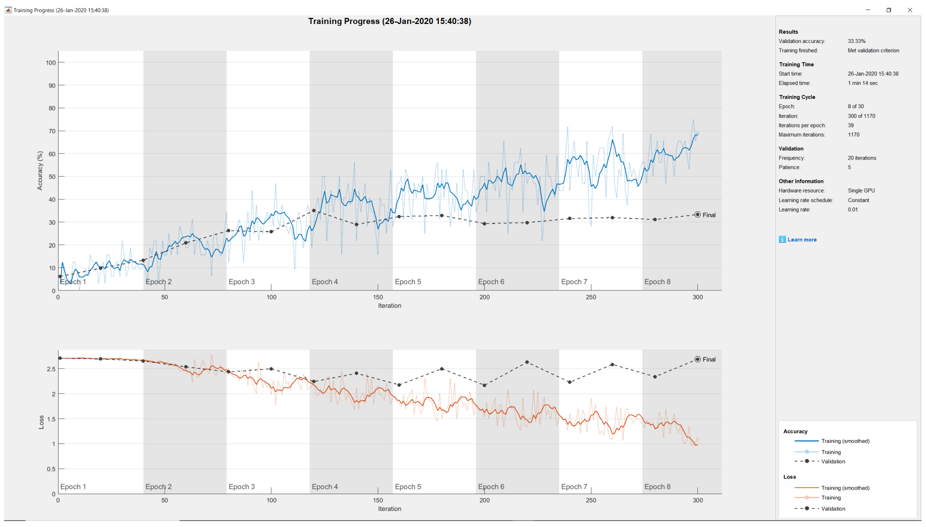
### Common Setup

For all steps and scripts there are some common features like:

- Input resizing: to match different CNN $1^{st}$ layer proprieties (in all three dimensions), I used "imresize", "repmat" and "augmentedImageDatastore" options.
- Input normalization: is almost always automatically done by $1^{st}$ layer.
- Data augmentation: to avoid overfitting and improve performance; I used translations, rotations, shear and reflection along vertical axe.
- Early stopping criteria: also to avoid overfitting, I used "MaxEpoch"($>$ 10 for my NN, 4 for transfer learning) and/or "Patience" (usually 6-8 iteration) on Validation Set Loss.
- Train-Validation split: 85%-15%.

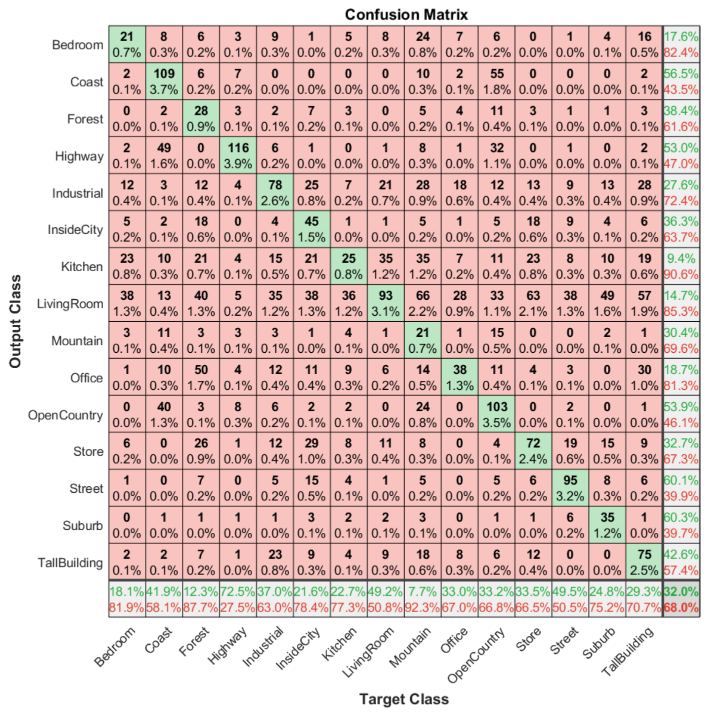
## Part 1

Just followed given instruction, only interesting part is recognize that (without using data augmentation) after a couple of epoch Validation Loss start to increase while Training Loss is still decreasing (same but reversed on accuracy score), meaning that the CNN is no more learning features but just overfitting on the Training Set.



| | |
|---|---|
| **Results** | |
| Validation accuracy: | 33.33% |
| Training finished: | Met validation criterion |
| **Training Time** | |
| Start time: | 26-Jan-2020 15:40:38 |
| Elapsed time: | 1 min 14 sec |
| **Training Cycle** | |
| Epoch: | 8 of 30 |
| Iteration: | 300 of 1170 |
| Iterations per epoch: | 39 |
| Maximum iterations: | 1170 |
| **Validation** | |
| Frequency: | 20 iterations |
| Patience: | 5 |
| **Other Information** | |
| Hardware resource: | Single GPU |
| Learning rate schedule: | Constant |
| Learning rate: | 0.01 |

Figure 1: Shallow NN


Confusion Matrix

Output Class / Target Class

| | Bedroom | Coast | Forest | Highway | Industrial | InsideCity | Kitchen | LivingRoom | Mountain | Office | OpenCountry | Store | Street | Suburb | TallBuilding | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bedroom | 21 0.7% | 8 0.3% | 6 0.2% | 3 0.1% | 9 0.3% | 1 0.0% | 5 0.2% | 8 0.3% | 24 0.8% | 7 0.2% | 6 0.2% | 0 0.0% | 1 0.0% | 4 0.1% | 16 0.5% | 17.6% 82.4% |
| Coast | 2 0.1% | 109 3.7% | 6 0.2% | 7 0.2% | 0 0.0% | 0 0.0% | 0 0.0% | 0 0.0% | 10 0.3% | 2 0.1% | 55 1.8% | 0 0.0% | 0 0.0% | 0 0.0% | 2 0.1% | 56.5% 43.5% |
| Forest | 0 0.0% | 2 0.1% | 28 0.9% | 3 0.1% | 2 0.1% | 7 0.2% | 3 0.1% | 0 0.0% | 5 0.2% | 4 0.1% | 11 0.4% | 3 0.1% | 1 0.0% | 1 0.0% | 3 0.1% | 38.4% 61.6% |
| Highway | 2 0.1% | 49 1.6% | 0 0.0% | 116 3.9% | 6 0.2% | 1 0.0% | 0 0.0% | 1 0.0% | 8 0.3% | 1 0.0% | 32 1.1% | 0 0.0% | 1 0.0% | 0 0.0% | 2 0.1% | 53.0% 47.0% |
| Industrial | 12 0.4% | 3 0.1% | 12 0.4% | 4 0.1% | 78 2.6% | 25 0.8% | 7 0.2% | 21 0.7% | 28 0.9% | 18 0.6% | 12 0.4% | 13 0.4% | 9 0.3% | 13 0.4% | 28 0.9% | 27.6% 72.4% |
| InsideCity | 5 0.2% | 2 0.1% | 18 0.6% | 0 0.0% | 4 0.1% | 45 1.5% | 1 0.0% | 1 0.0% | 5 0.2% | 1 0.0% | 5 0.2% | 18 0.6% | 9 0.3% | 4 0.1% | 6 0.2% | 36.3% 63.7% |
| Kitchen | 23 0.8% | 10 0.3% | 21 0.7% | 4 0.1% | 15 0.5% | 21 0.7% | 25 0.8% | 35 1.2% | 35 1.2% | 7 0.2% | 11 0.4% | 23 0.8% | 8 0.3% | 10 0.3% | 19 0.6% | 9.4% 90.6% |
| LivingRoom | 38 1.3% | 13 0.4% | 40 1.3% | 5 0.2% | 35 1.2% | 38 1.3% | 36 1.2% | 93 3.1% | 66 2.2% | 28 0.9% | 33 1.1% | 63 2.1% | 38 1.3% | 49 1.6% | 57 1.9% | 14.7% 85.3% |
| Mountain | 3 0.1% | 11 0.4% | 3 0.1% | 3 0.1% | 3 0.1% | 1 0.0% | 4 0.1% | 1 0.0% | 21 0.7% | 1 0.0% | 15 0.5% | 0 0.0% | 0 0.0% | 2 0.1% | 1 0.0% | 30.4% 69.6% |
| Office | 1 0.0% | 10 0.3% | 50 1.7% | 4 0.1% | 12 0.4% | 11 0.4% | 9 0.3% | 6 0.2% | 14 0.5% | 38 1.3% | 11 0.4% | 4 0.1% | 3 0.1% | 0 0.0% | 30 1.0% | 18.7% 81.3% |
| OpenCountry | 0 0.0% | 40 1.3% | 3 0.1% | 8 0.3% | 6 0.2% | 2 0.1% | 2 0.1% | 0 0.0% | 24 0.8% | 0 0.0% | 103 3.5% | 0 0.0% | 2 0.1% | 0 0.0% | 1 0.0% | 53.9% 46.1% |
| Store | 6 0.2% | 0 0.0% | 26 0.9% | 1 0.0% | 12 0.4% | 29 1.0% | 8 0.3% | 11 0.4% | 8 0.3% | 0 0.0% | 4 0.1% | 72 2.4% | 19 0.6% | 15 0.5% | 9 0.3% | 32.7% 67.3% |
| Street | 1 0.0% | 0 0.0% | 7 0.2% | 0 0.0% | 5 0.2% | 15 0.5% | 4 0.1% | 1 0.0% | 5 0.2% | 0 0.0% | 5 0.2% | 6 0.2% | 95 3.2% | 8 0.3% | 6 0.2% | 60.1% 39.9% |
| Suburb | 0 0.0% | 1 0.0% | 1 0.0% | 1 0.0% | 1 0.0% | 3 0.1% | 2 0.1% | 2 0.1% | 3 0.1% | 0 0.0% | 1 0.0% | 1 0.0% | 6 0.2% | 35 1.2% | 1 0.0% | 60.3% 39.7% |
| TallBuilding | 2 0.1% | 2 0.1% | 7 0.2% | 1 0.0% | 23 0.8% | 9 0.3% | 4 0.1% | 9 0.3% | 18 0.6% | 8 0.3% | 6 0.2% | 12 0.4% | 0 0.0% | 0 0.0% | 75 2.5% | 42.6% 57.4% |
| | 18.1% 81.9% | 41.9% 58.1% | 12.3% 87.7% | 72.5% 27.5% | 37.0% 63.0% | 21.6% 78.4% | 22.7% 77.3% | 49.2% 50.8% | 7.7% 92.3% | 33.0% 67.0% | 33.2% 66.8% | 33.5% 66.5% | 49.5% 50.5% | 24.8% 75.2% | 29.3% 70.7% | 32.0% 68.0% |

Figure 2: Shallow NN

The overall validation accuracy is around 33% vs. a training accuracy reaching the 60% (clearly overfitting). Test accuracy is around 32%.

# Part 2

For that part the goal was to improve the CNN with some different techniques.

## Data-augmentation

For data augmentation I decided to use Shear, Rotation, Reflection and Translations, with following parameters:

```
imageAugmenter = imageDataAugmenter( ...
        'RandXShear',[-6,6], ...
        'RandYShear',[-6,6], ...
        'RandRotation',[-10,10], ...
        'RandXReflection', true, ...
        'RandXTranslation',[-6 6], ...
        'RandYTranslation',[-6 6]);

augimds_Train = augmentedImageDatastore(imageSize,
    imdsTrain, 'DataAugmentation',imageAugmenter);
```

Given the tiny input size i thought was better to limit the parameters to a 10% of the images dimensions and keep rotations small (intuitively to kind of the match human error on taking pictures and to emphasize the fact that in our problem the orientation is quite important).

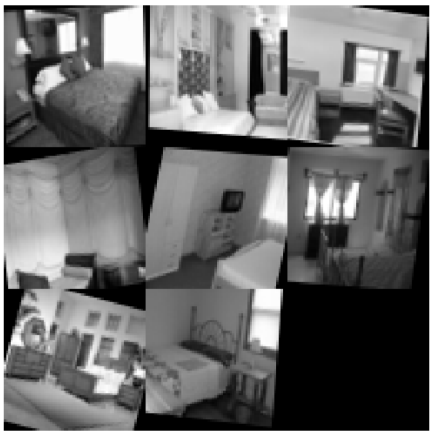
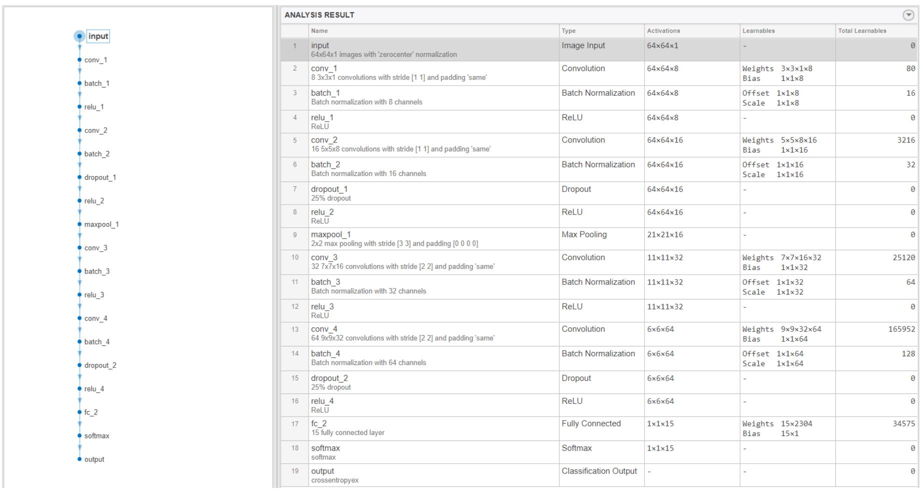


Figure 3: Augmented bedroom batch

## More Layers

I added some layers:

- Convolutional: to improve the capability of features extraction.
- Batch-Normalization after every CL: Speed-up the training by normalizing activators and mitigate sensitivity to initialization randomness.
- Dropout: improve generalization and prevent overfitting by randomly setting to zero values with the given probability.

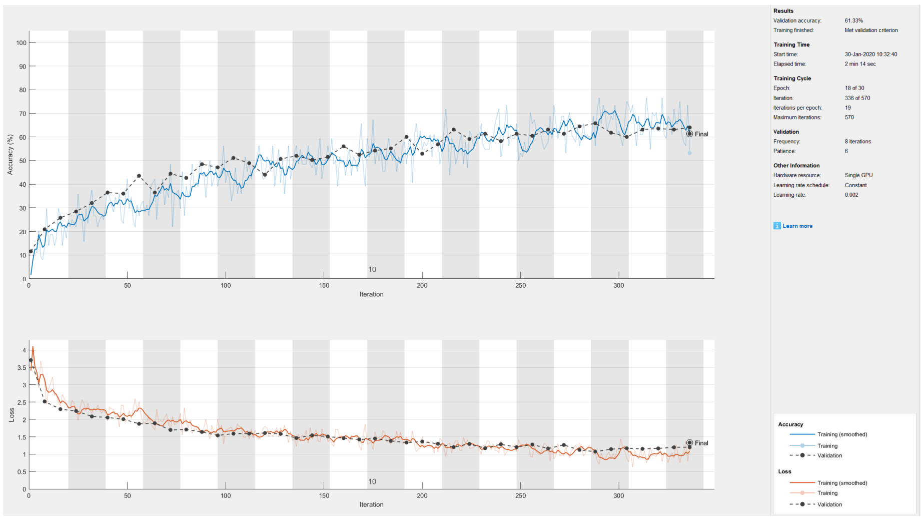


ANALYSIS RESULT

|  | Name | Type | Activations | Learnables | Total Learnables |
|---|---|---|---|---|---|
| 1 | input<br>64x64x1 images with 'zerocenter' normalization | Image Input | 64×64×1 | - | 0 |
| 2 | conv_1<br>8 3x3x1 convolutions with stride [1 1] and padding 'same' | Convolution | 64×64×8 | Weights 3×3×1×8<br>Bias 1×1×8 | 80 |
| 3 | batch_1<br>Batch normalization with 8 channels | Batch Normalization | 64×64×8 | Offset 1×1×8<br>Scale 1×1×8 | 16 |
| 4 | relu_1<br>ReLU | ReLU | 64×64×8 | - | 0 |
| 5 | conv_2<br>16 5x5x8 convolutions with stride [1 1] and padding 'same' | Convolution | 64×64×16 | Weights 5×5×8×16<br>Bias 1×1×16 | 3216 |
| 6 | batch_2<br>Batch normalization with 16 channels | Batch Normalization | 64×64×16 | Offset 1×1×16<br>Scale 1×1×16 | 32 |
| 7 | dropout_1<br>25% dropout | Dropout | 64×64×16 | - | 0 |
| 8 | relu_2<br>ReLU | ReLU | 64×64×16 | - | 0 |
| 9 | maxpool_1<br>2x2 max pooling with stride [3 3] and padding [0 0 0 0] | Max Pooling | 21×21×16 | - | 0 |
| 10 | conv_3<br>32 7x7x16 convolutions with stride [2 2] and padding 'same' | Convolution | 11×11×32 | Weights 7×7×16×32<br>Bias 1×1×32 | 25120 |
| 11 | batch_3<br>Batch normalization with 32 channels | Batch Normalization | 11×11×32 | Offset 1×1×32<br>Scale 1×1×32 | 64 |
| 12 | relu_3<br>ReLU | ReLU | 11×11×32 | - | 0 |
| 13 | conv_4<br>64 9x9x32 convolutions with stride [2 2] and padding 'same' | Convolution | 6×6×64 | Weights 9×9×32×64<br>Bias 1×1×64 | 165952 |
| 14 | batch_4<br>Batch normalization with 64 channels | Batch Normalization | 6×6×64 | Offset 1×1×64<br>Scale 1×1×64 | 128 |
| 15 | dropout_2<br>25% dropout | Dropout | 6×6×64 | - | 0 |
| 16 | relu_4<br>ReLU | ReLU | 6×6×64 | - | 0 |
| 17 | fc_2<br>15 fully connected layer | Fully Connected | 1×1×15 | Weights 15×2304<br>Bias 15×1 | 34575 |
| 18 | softmax<br>softmax | Softmax | 1×1×15 | - | 0 |
| 19 | output<br>crossentropyex | Classification Output | - | - | 0 |

Figure 4: Improved NN Structure

## Results



**Results**

| | |
|---|---|
| Validation accuracy: | 61.33% |
| Training finished: | Met validation criterion |

**Training Time**

| | |
|---|---|
| Start time: | 30-Jan-2020 10:32:40 |
| Elapsed time: | 2 min 14 sec |

**Training Cycle**

| | |
|---|---|
| Epoch: | 18 of 30 |
| Iteration: | 336 of 570 |
| Iterations per epoch: | 19 |
| Maximum iterations: | 570 |

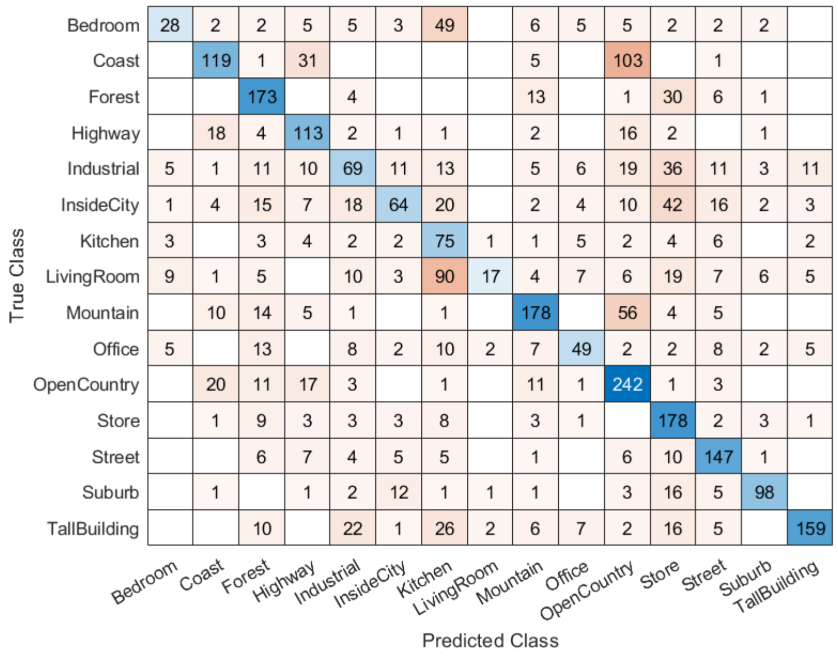
**Validation**

| | |
|---|---|
| Frequency: | 8 iterations |
| Patience: | 6 |

**Other Information**

| | |
|---|---|
| Hardware resource: | Single GPU |
| Learning rate schedule: | Constant |
| Learning rate: | 0.002 |

Figure 5: Improved NN


True Class
Predicted Class

| | Bedroom | Coast | Forest | Highway | Industrial | InsideCity | Kitchen | LivingRoom | Mountain | Office | OpenCountry | Store | Street | Suburb | TallBuilding |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bedroom | 28 | 2 | 2 | 5 | 5 | 3 | 49 | | 6 | 5 | 5 | 2 | 2 | 2 | |
| Coast | | 119 | 1 | 31 | | | | | 5 | | 103 | | 1 | | |
| Forest | | | 173 | | 4 | | | | 13 | | 1 | 30 | 6 | 1 | |
| Highway | | 18 | 4 | 113 | 2 | 1 | 1 | | 2 | | 16 | 2 | | 1 | |
| Industrial | 5 | 1 | 11 | 10 | 69 | 11 | 13 | | 5 | 6 | 19 | 36 | 11 | 3 | 11 |
| InsideCity | 1 | 4 | 15 | 7 | 18 | 64 | 20 | | 2 | 4 | 10 | 42 | 16 | 2 | 3 |
| Kitchen | 3 | | 3 | 4 | 2 | 2 | 75 | 1 | 1 | 5 | 2 | 4 | 6 | | 2 |
| LivingRoom | 9 | 1 | 5 | | 10 | 3 | 90 | 17 | 4 | 7 | 6 | 19 | 7 | 6 | 5 |
| Mountain | | 10 | 14 | 5 | 1 | | 1 | | 178 | | 56 | 4 | 5 | | |
| Office | 5 | | 13 | | 8 | 2 | 10 | 2 | 7 | 49 | 2 | 2 | 8 | 2 | 5 |
| OpenCountry | | 20 | 11 | 17 | 3 | | 1 | | 11 | 1 | 242 | 1 | 3 | | |
| Store | | 1 | 9 | 3 | 3 | 3 | 8 | | 3 | 1 | | 178 | 2 | 3 | 1 |
| Street | | | 6 | 7 | 4 | 5 | 5 | | 1 | | 6 | 10 | 147 | 1 | |
| Suburb | | 1 | | 1 | 2 | 12 | 1 | 1 | 1 | | 3 | 16 | 5 | 98 | |
| TallBuilding | | | 10 | | 22 | 1 | 26 | 2 | 6 | 7 | 2 | 16 | 5 | | 159 |

Figure 6: Improved NN

The overall validation accuracy is around 61%, almost equal to training one, test accuracy is around 57%. Notice how all the new features prevent the CNN to overfit the training set, reaching a kind of plateau with train and validation losses lines really close to each over.

## Ensemble of NN

I've also tried to use an ensemble of 10 of the previous NN.

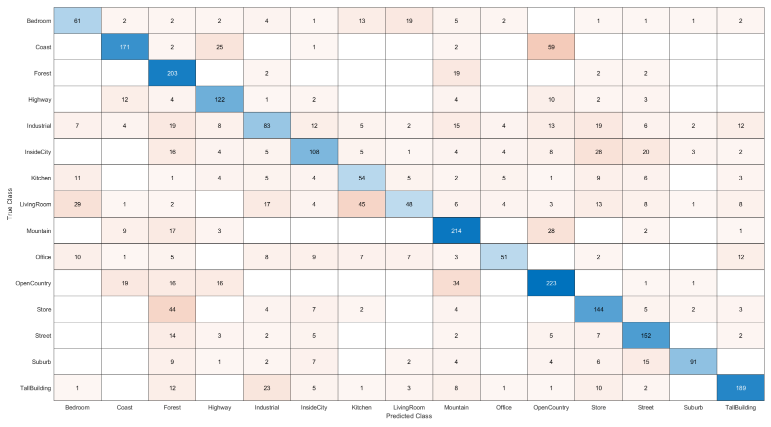


Figure 7: Ensemble of improved NN

I've reached a test accuracy a bit over 64% using the arithmetic means of the output classification score from each CNN.

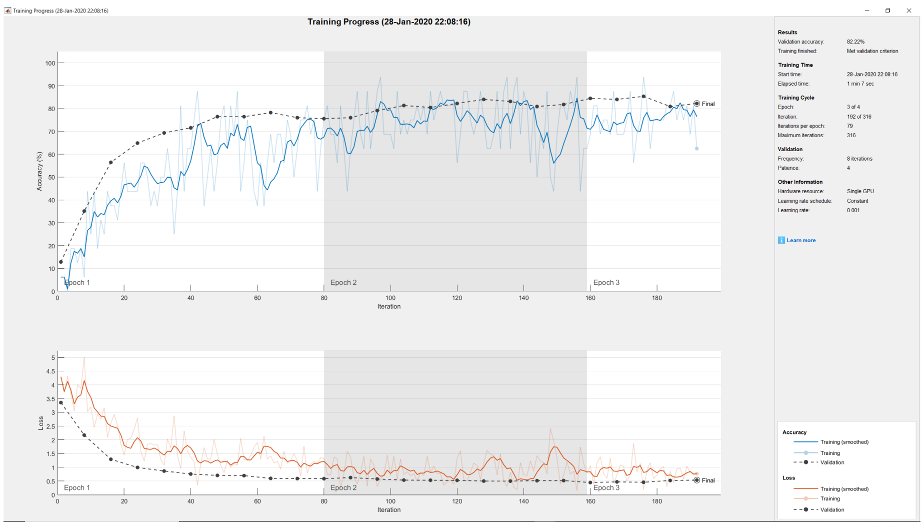
## Part 3

Lastly was required to adopt a Transfer Learning approach importing using trained CNNs. I used almost the same data augmentation techniques as before, I just added the gray2rgb option to match imported networks input size. There are two approach for this task:

- Replace last Full-Connected layer with another one (with the right dimension for our classification problem) and then fine-tune the derived CNN.
- Get activators from a Convolutional, ReLU or Pooling layer and then train a SVM to get classification.

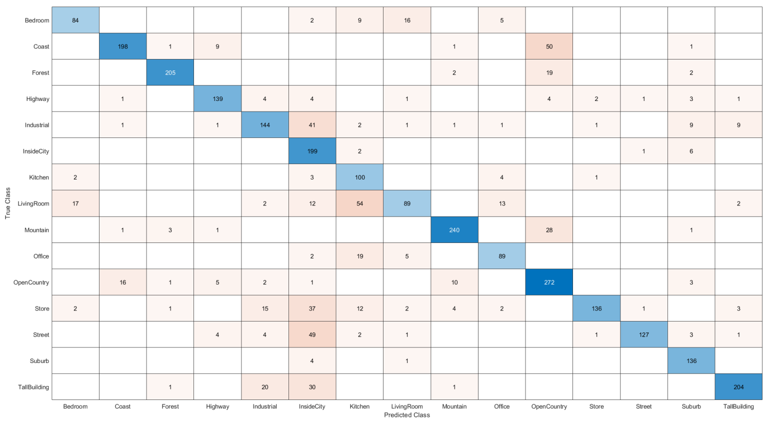
### Replace

I tried first approach with AlexNet and VGG, for AlexNet I tried both freezing all layers except the new one and no freeze but with 0.1x of learning rate and 20x LearnFactor on last layer. For VGG the only possible option due HW limitation was the first one.



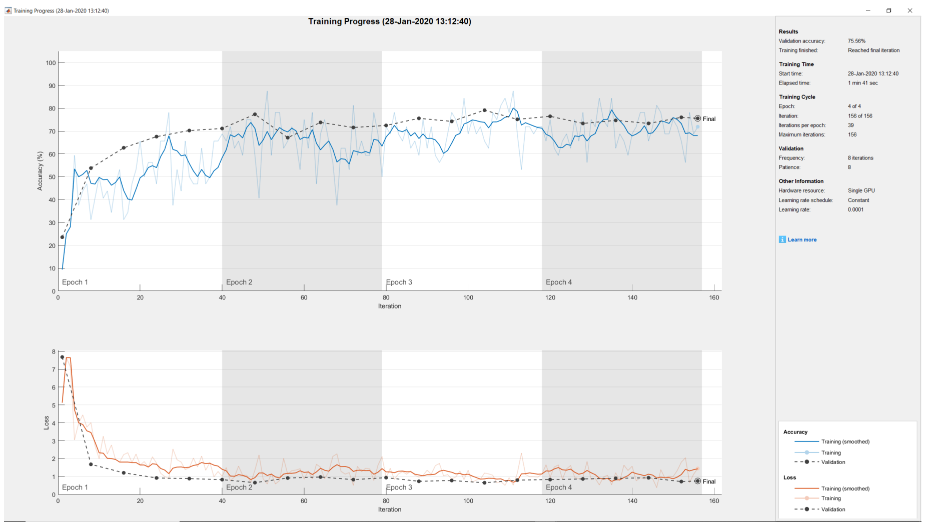
| | |
|---|---|
| **Results** | |
| Validation accuracy: | 82.22% |
| Training finished: | Met validation criterion |
| **Training Time** | |
| Start time: | 28-Jan-2020 22:08:16 |
| Elapsed time: | 1 min 7 sec |
| **Training Cycle** | |
| Epoch: | 3 of 4 |
| Iteration: | 192 of 316 |
| Iterations per epoch: | 79 |
| Maximum iterations: | 316 |
| **Validation** | |
| Frequency: | 8 iterations |
| Patience: | 4 |
| **Other Information** | |
| Hardware resource: | Single GPU |
| Learning rate schedule: | Constant |
| Learning rate: | 0.001 |

Figure 8: AlexNet Freeze



| True Class \ Predicted Class | Bedroom | Coast | Forest | Highway | Industrial | InsideCity | Kitchen | LivingRoom | Mountain | Office | OpenCountry | Store | Street | Suburb | TallBuilding |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bedroom | 84 | | | | | 2 | 9 | 16 | | 5 | | | | | |
| Coast | | 198 | 1 | 9 | | | | | 1 | | 50 | | | 1 | |
| Forest | | | 205 | | | | | | 2 | | 19 | | | 2 | |
| Highway | | 1 | | 139 | 4 | 4 | | 1 | | | 4 | 2 | 1 | 3 | 1 |
| Industrial | | 1 | | 1 | 144 | 41 | 2 | 1 | 1 | 1 | | 1 | | 9 | 9 |
| InsideCity | | | | | | 199 | 2 | | | | | | 1 | 6 | |
| Kitchen | 2 | | | | | 3 | 100 | | | 4 | | 1 | | | |
| LivingRoom | 17 | | | | 2 | 12 | 54 | 89 | | 13 | | | | | 2 |
| Mountain | | 1 | 3 | 1 | | | | | 240 | | 28 | | | 1 | |
| Office | | | | | | 2 | 19 | 5 | | 89 | | | | | |
| OpenCountry | | 16 | 1 | 5 | 2 | 1 | | | 10 | | 272 | | | 3 | |
| Store | 2 | | 1 | | 15 | 37 | 12 | 2 | 4 | 2 | | 136 | 1 | | 3 |
| Street | | | | 4 | 4 | 49 | 2 | 1 | | | | 1 | 127 | 3 | 1 |
| Suburb | | | | | | 4 | | 1 | | | | | | 136 | |
| TallBuilding | | | 1 | | 20 | 30 | | | 1 | | | | | | 204 |

Figure 9: AlexNet Freeze



**Results**
Validation accuracy: 75.56%
Training finished: Reached final iteration

**Training Time**
Start time: 28-Jan-2020 13:12:40
Elapsed time: 1 min 41 sec

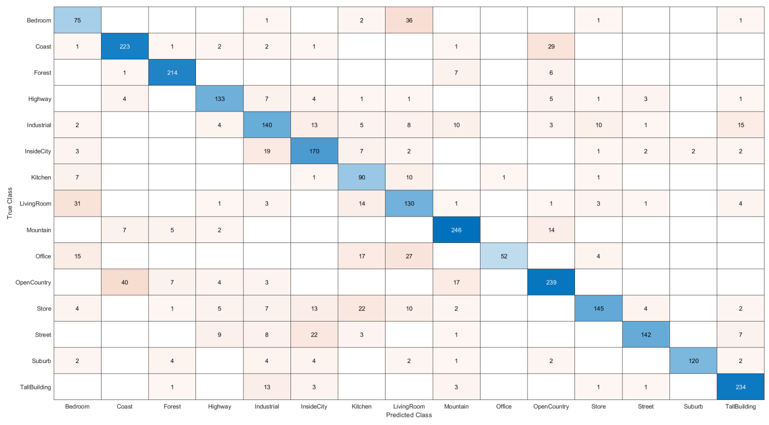
**Training Cycle**
Epoch: 4 of 4
Iteration: 156 of 156
Iterations per epoch: 39
Maximum iterations: 156

**Validation**
Frequency: 8 iterations
Patience: 8

**Other Information**
Hardware resource: Single GPU
Learning rate schedule: Constant
Learning rate: 0.0001

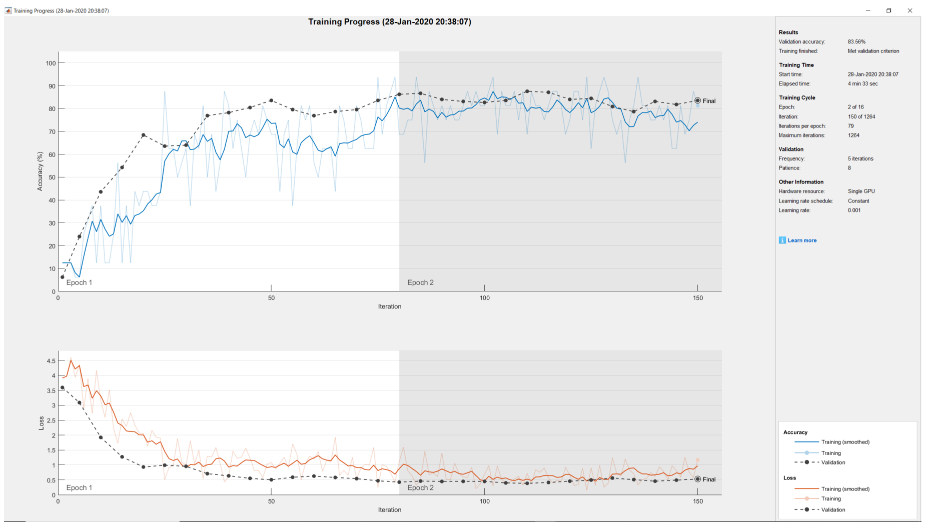
Learn more

Figure 10: AlexNet No Freeze



| True Class \ Predicted Class | Bedroom | Coast | Forest | Highway | Industrial | InsideCity | Kitchen | LivingRoom | Mountain | Office | OpenCountry | Store | Street | Suburb | TallBuilding |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bedroom | 75 | | | | 1 | | 2 | 36 | | | | 1 | | | 1 |
| Coast | 1 | 223 | 1 | 2 | 2 | 1 | | | 1 | | 29 | | | | |
| Forest | | 1 | 214 | | | | | | 7 | | 6 | | | | |
| Highway | | 4 | | 133 | 7 | 4 | 1 | 1 | | | 5 | 1 | 3 | | 1 |
| Industrial | 2 | | | 4 | 140 | 13 | 5 | 8 | 10 | | 3 | 10 | 1 | | 15 |
| InsideCity | 3 | | | | 19 | 170 | 7 | 2 | | | | 1 | 2 | 2 | 2 |
| Kitchen | 7 | | | | | 1 | 90 | 10 | | 1 | | 1 | | | |
| LivingRoom | 31 | | | 1 | 3 | | 14 | 130 | 1 | | 1 | 3 | 1 | | 4 |
| Mountain | | 7 | 5 | 2 | | | | | 246 | | 14 | | | | |
| Office | 15 | | | | | | 17 | 27 | | 52 | | 4 | | | |
| OpenCountry | | 40 | 7 | 4 | 3 | | | | 17 | | 239 | | | | |
| Store | 4 | | 1 | 5 | 7 | 13 | 22 | 10 | 2 | | | 145 | 4 | | 2 |
| Street | | | | 9 | 8 | 22 | 3 | | 1 | | | | 142 | | 7 |
| Suburb | 2 | | 4 | | 4 | 4 | | 2 | 1 | | 2 | | | 120 | 2 |
| TallBuilding | | | 1 | | 13 | 3 | | | 3 | | | 1 | 1 | | 234 |

Figure 11: AlexNet No Freeze



**Results**

| | |
|---|---|
| Validation accuracy: | 83.56% |
| Training finished: | Met validation criterion |

**Training Time**

| | |
|---|---|
| Start time: | 28-Jan-2020 20:38:07 |
| Elapsed time: | 4 min 33 sec |

**Training Cycle**

| | |
|---|---|
| Epoch: | 2 of 16 |
| Iteration: | 150 of 1264 |
| Iterations per epoch: | 79 |
| Maximum iterations: | 1264 |

**Validation**

| | |
|---|---|
| Frequency: | 5 iterations |
| Patience: | 8 |

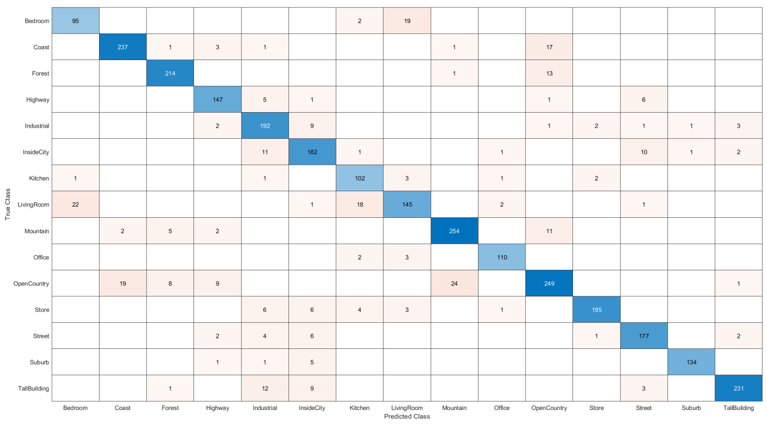
**Other Information**

| | |
|---|---|
| Hardware resource: | Single GPU |
| Learning rate schedule: | Constant |
| Learning rate: | 0.001 |

Figure 12: VGG

## Supported Vector Machines

For second approach I used Resnet18 and Error-Correcting Output Codes Model for training the SVM classifier. Given the structure of this network I decide to extract the activators from last GlobalPooling layer which become the features vector of every image (512 dim. feature space).



| True Class \ Predicted Class | Bedroom | Coast | Forest | Highway | Industrial | InsideCity | Kitchen | LivingRoom | Mountain | Office | OpenCountry | Store | Street | Suburb | TallBuilding |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Bedroom | 95 | | | | | | 2 | 19 | | | | | | | |
| Coast | | 237 | 1 | 3 | 1 | | | | 1 | | 17 | | | | |
| Forest | | | 214 | | | | | | 1 | | 13 | | | | |
| Highway | | | | 147 | 5 | 1 | | | | | 1 | | 6 | | |
| Industrial | | | | 2 | 192 | 9 | | | | | 1 | 2 | 1 | 1 | 3 |
| InsideCity | | | | | 11 | 182 | 1 | | | 1 | | | 10 | 1 | 2 |
| Kitchen | 1 | | | | 1 | | 102 | 3 | | 1 | | 2 | | | |
| LivingRoom | 22 | | | | | 1 | 18 | 145 | | 2 | | | 1 | | |
| Mountain | | 2 | 5 | 2 | | | | | 254 | | 11 | | | | |
| Office | | | | | | | 2 | 3 | | 110 | | | | | |
| OpenCountry | | 19 | 8 | 9 | | | | | 24 | | 249 | | | | 1 |
| Store | | | | | 6 | 6 | 4 | 3 | | 1 | | 195 | | | |
| Street | | | | 2 | 4 | 6 | | | | | | 1 | 177 | | 2 |
| Suburb | | | | 1 | 1 | 5 | | | | | | | | 134 | |
| TallBuilding | | | 1 | | 12 | 9 | | | | | | | 3 | | 231 |

Figure 13: SMV

## Comment on results of part 3

All approaches went beyond 75% accuracy, with the SVM almost crossing 90% accuracy test score. Notice also that no overfitting is present (which can be a common problem when dealing with transfer learning due to the imbalance between the huge prior training set and the new one) this is mostly thanks to data augmentation applied on training set and early-stopping criterion on training routine.
For second approach i could also extract features form one of the last CL, but I'd get a 25k dimensions feature vector, which is too big for our problem in my opinion.