IMAGE CLASSIFICATION OF OUTDOOR SCENES WITH CONVOLUTIONAL NEURAL NETWORKS

ADAM YOUNG

PROBLEM STATEMENT, DATA AND PREPROCESSING

PROBLEM STATEMENT

- Given an input image, classify it into one of 6 categories:

Buildings

Mountains

Forests

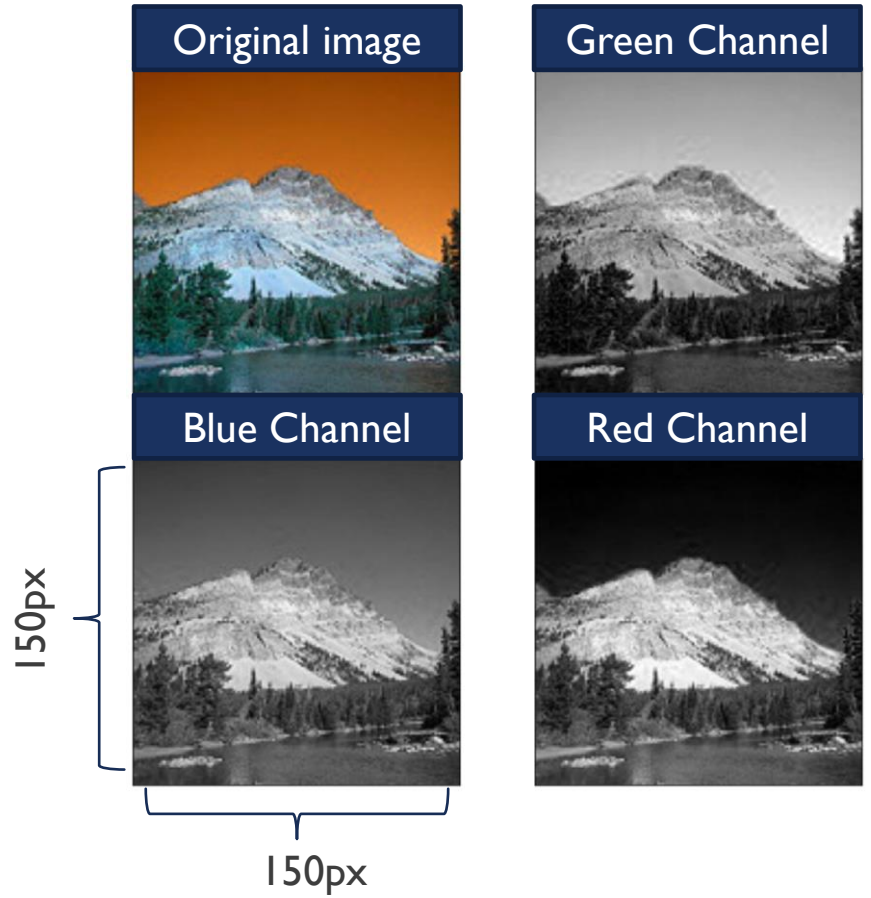
Sea

Streets

Glaciers

DATA DESCRIPTION AND IMPORT

- Data comes from the Intel Image Classification Dataset
  - Training set of 14,034 images, test set of 3,000 images
- Data import process:
  - Loop through directory and load each image
  - Resize image to 150 by 150 pixels
  - Convert image to array with 3 channels
  - Scale values between 0 and 1
  - Append image and its class to training or testing dataset
- Final training data:
  - Array of images with shape (14034, 150, 150, 3)
  - Array of classes with shape (14034, 1)



APPROACH

- Build and test a basic Convolutional Neural Network (CNN)
- Tune the basic CNN's performance
  - Different architectures
  - Image augmentation
  - Training for additional epochs
- Implement a transfer-learning based model and compare results with the basic CNN

JUSTIFICATION FOR USING A CNN

- CNNs are great for image classification tasks because of their ability to learn complex relationships within spatially related data
- There are 3 main components that allow a CNN to detect these complex relationships:

Convolutional Layers

- Learn and detect features by sliding filters across the input image
- These features generally increase in complexity the deeper the convolutional layer is in the network

Pooling Layers

- Down-sample the feature maps output by the convolutional layers
- Helps model become more robust to slight changes in the position of features within the input image

Fully Connected Layers

- Flatten the final feature maps and help the model connect the dots between the output features and the predicted classes

VISUALIZATION OF A CNN

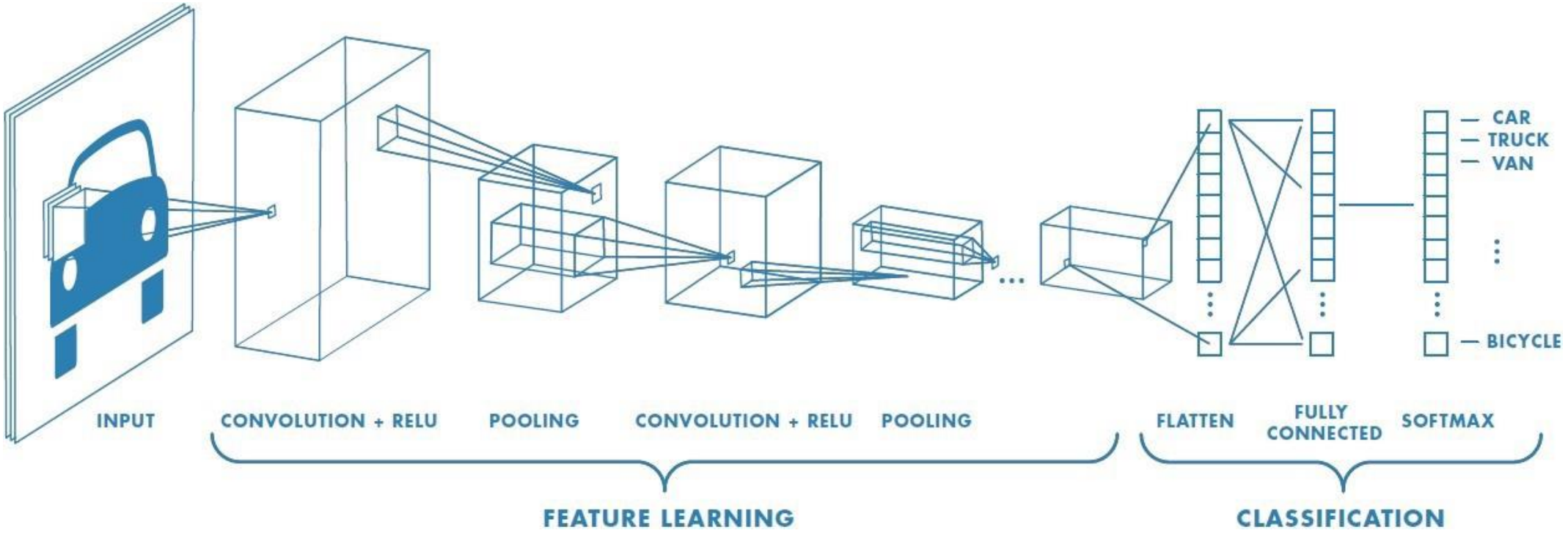


Image credit: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

MODELING

INITIAL RESULTS

- The first model tested was a simple CNN with three ReLu-activated convolutional layers, each followed by a max pooling layer, and then a dense layer at the end.
- After training for 20 epochs these were the results on the validation data:

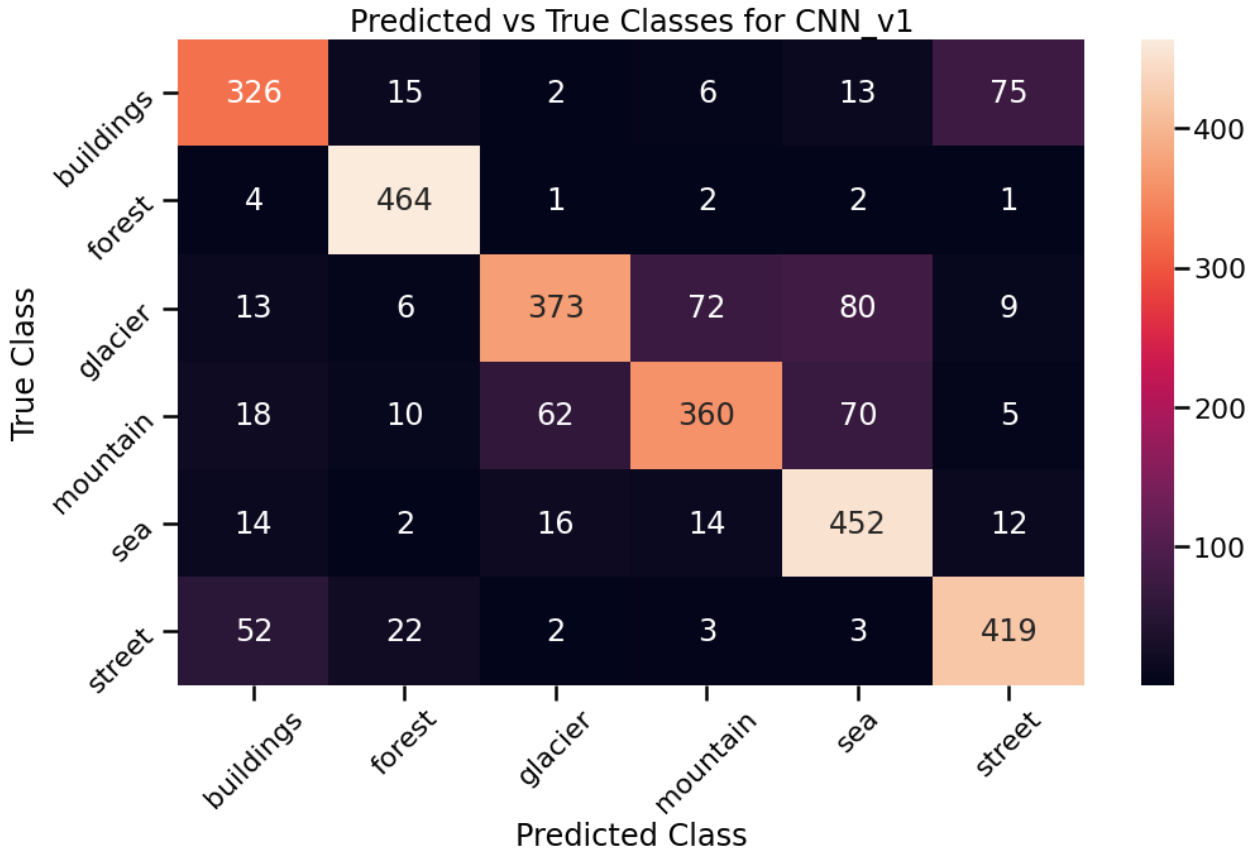
Accuracy	Log-loss	Macro-Average F1 Score
79.80%	0.9928	0.7969

INITIAL MODEL CONFUSION MATRIX

- The overall accuracy of the initial model was not too bad, but it's important to dig into the confusion matrix and see which areas the model is struggling with the most

- The basic model appears to struggle with:
  - Confusing streets with buildings
  - Confusing mountains with glaciers
  - Confusing glaciers and mountains with the sea

- These errors make logical sense because:
  - Streets often have buildings in the background
  - Glaciers are essentially just icy mountains
  - A sea's coastline can look like a mountain or glacier



MODEL IMPROVEMENT DETAILS

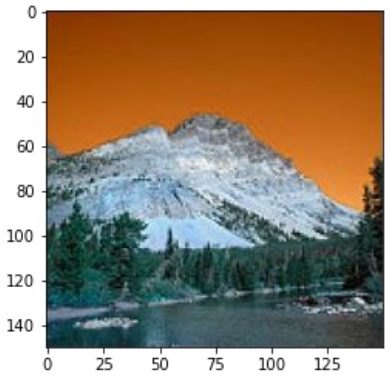
- The table below summarizes the different models that were tested while trying to improve performance:

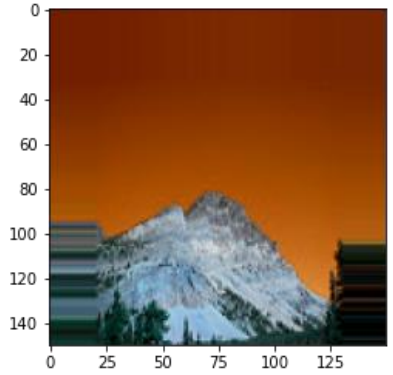
Model	Description	Val. Accuracy
1	20 epochs, max pooling, no image augmentation	79.80%
2	20 epochs, max pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2)	86.30%
3	20 epochs, max pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2, height shift = 0.2, rotation range=0.2)	84.90%
4	20 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4)	87.40%
5	20 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4, height shift = 0.4, rotation range=0.4)	84.13%
6	100 epochs, max pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2)	86.50%
7	20 epochs, average pooling, image augmentation (width shift=0.2, horizontal flip=True, zoom range=0.2, shear range=0.2)	85.20%
8	100 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4)	87.47%
9	100 epochs, max pooling, image augmentation (width shift=0.4, horizontal flip=True, zoom range=0.4, shear range=0.4, channel shift=0.4)	87.03%

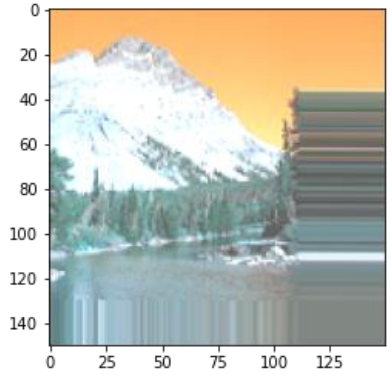
INTUITION BEHIND IMAGE AUGMENTATION

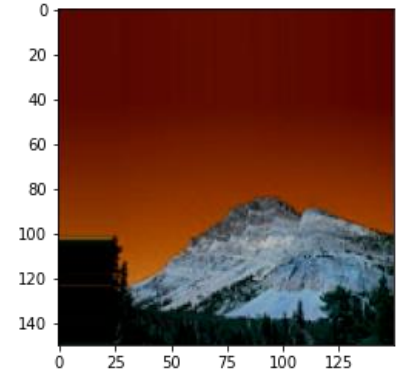
- The biggest jump in performance was obtained through a process called image augmentation, which involves "augmenting" the images in the training set in order to generate pseudo-new data.
- By exposing the model to additional instances of shifted, rotated and cropped versions of the training images, the model is more likely to identify the important features within the images, allowing it to generalize better to new data
- Be sure to only use augmentations that make sense for your data. For example, would it make sense to flip a mountain upside down?

Original Image	Darkened, Shifted and Zoomed	Brightened and Shifted	Darkened, Flipped and Zoomed





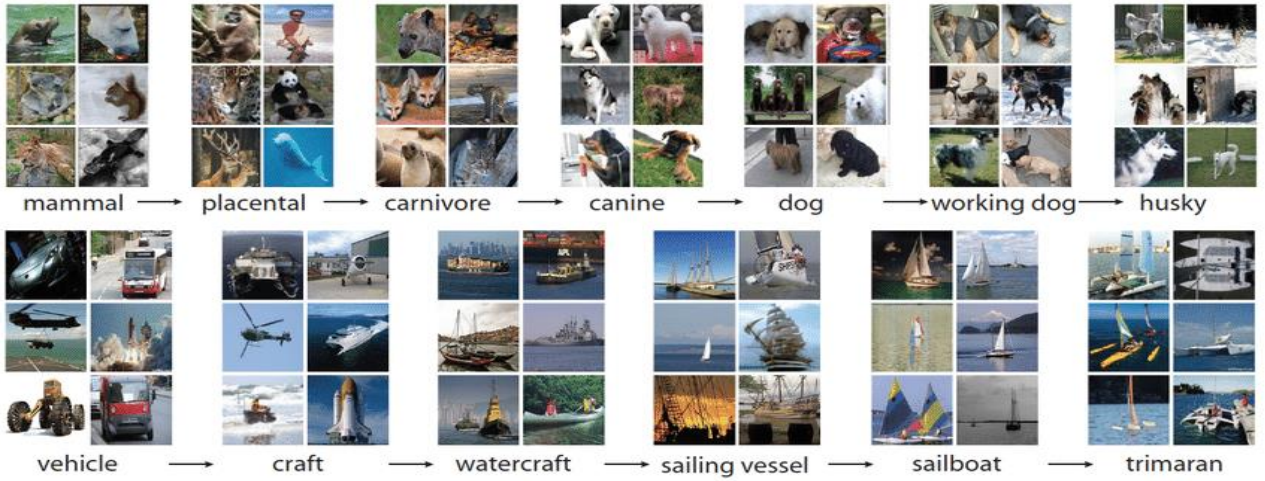




TRANSFER LEARNING

INTUITION BEHIND TRANSFER LEARNING

- While tuning the basic model, I didn't make changes to the overall architecture of the model
- For image classification tasks, the best model architecture is usually one that is stolen from someone else
- Researchers and tech companies have developed extremely complicated network architectures and trained them for thousands of hours on large datasets like ImageNet (which has over 14 million images and over 21 thousand classes)
- With transfer learning, we can take advantage of the features that these networks have learned and apply them to our own classification tasks **(so long as the images are reasonably similar)**
- The ImageNet database contains a wide array of image types, including many outdoor images, so it should be suitable for the outdoor image classification task
- Some example ImageNet data classes:



TRANSFER LEARNING RESULTS

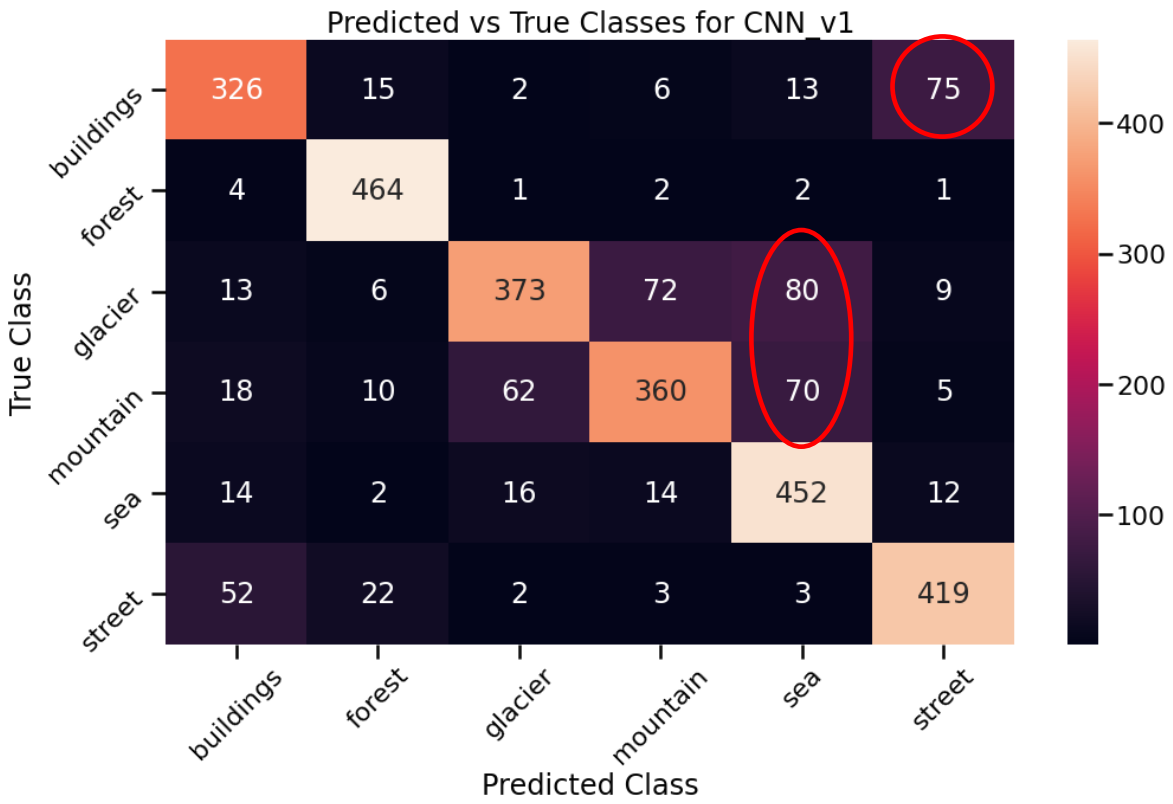
- 3 well-known architectures were tested: Xception, ResNet50 and DenseNet

Model	Val. Accuracy
Xception	88.60%
DenseNet	89.63%
ResNet	89.27%

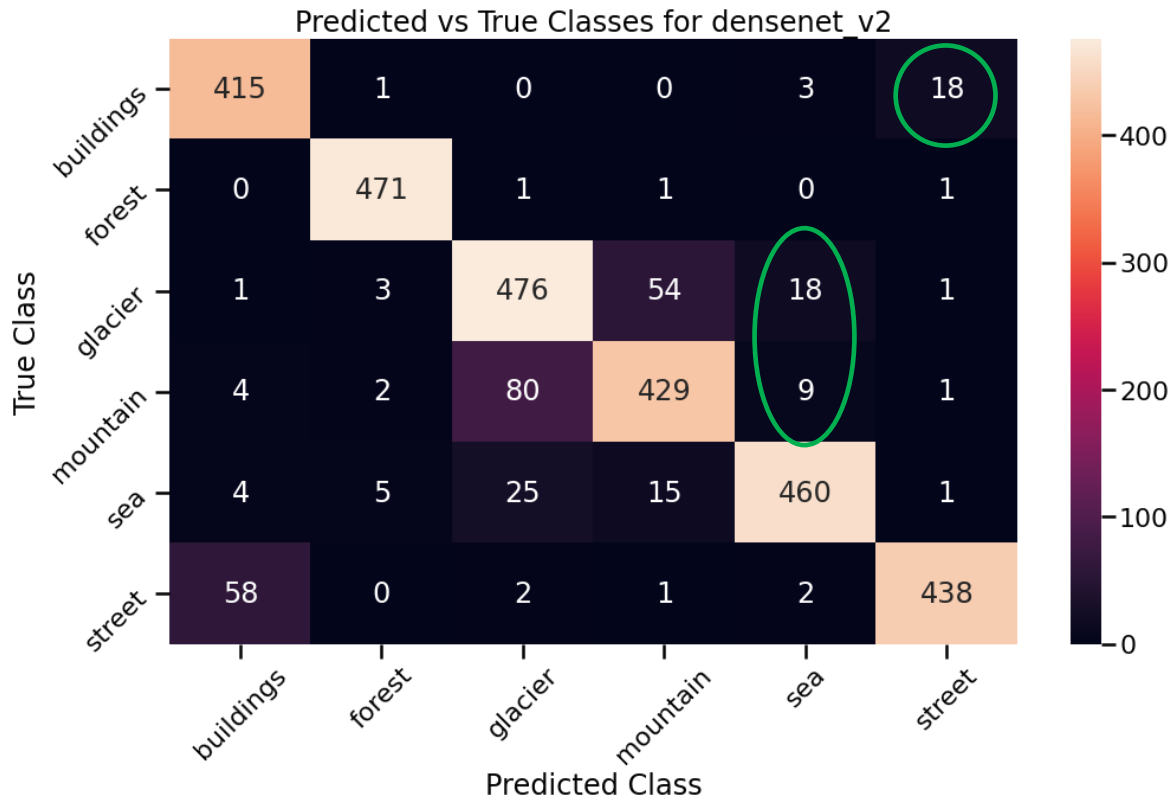
- DenseNet was the best performing model, resulting in a 2.16% increase in accuracy compared to the 87.47% that was achieved with the basic CNN
- By taking a pre-trained architecture and adding a dense layer on top, the model is able to decide which of the filters and features are useful, and then apply them to the classification task at hand

REVISITING THE CONFUSION MATRIX

Initial Model
Accuracy = 79.80%



DenseNet
Accuracy = 89.63%



TAKEAWAYS

- Basic CNN model accuracy peaked at 87.47%
  - Image augmentation is key for developing accurate models on smaller datasets
- Transfer learning improved the model accuracy to 89.63%
  - Work smarter, not harder

FUTURE IMPROVEMENTS

- Transfer learning performance can be pushed even further by un-freezing the top layers of the network and retraining them on your dataset
- Given additional time and computational resources, it would be interesting to explore what accuracy can be achieved with this method