Natural Scenes Image Classification

Machine Learning 1 (Fall 2021)

George Washington University

Group 1

Hassen, Adel

Teinfalt, Lydia

Vasquez-Perez, Pedro

Introduction

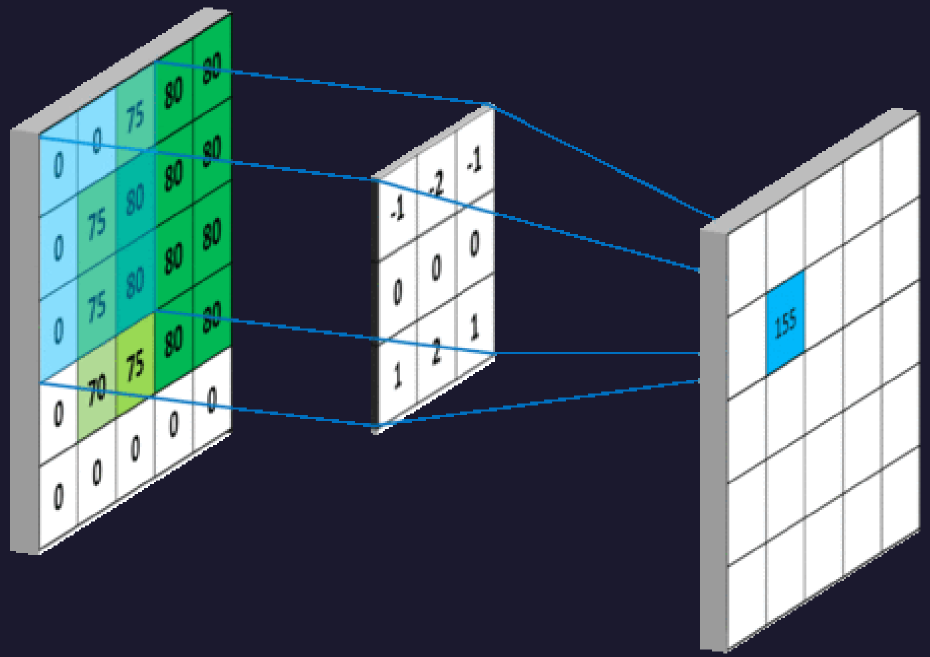
- Dataset
- Convolutional Neural Network
- Preprocessing
- Models
- Results

Dataset

- Kaggle dataset "Intel Image Classification" of natural scenes around the world. Images are in color and contain 150x150 pixels
- The dataset separated train and test data into main directories and placed each image in a subdirectory corresponding to its class

Total Number of Images	17,034
• Training	14,034
• Test	3,000

Classes	Number
• Building	0
• Forest	1
• Glacier	2
• Mountain	3
• Sea	4
• Street	5



Convoluted Neural Network

- Feedforward Neural Network (FNN) is not scalable for image processing. Perceptron required for each pixel of image
- FNN sensitive to variations in image whereas CNN can handle variations in image
- CNN is superior model to FNN in terms of accuracy and speed because it is designed for computer vision

Data Visualization

- What do images from each class look like?

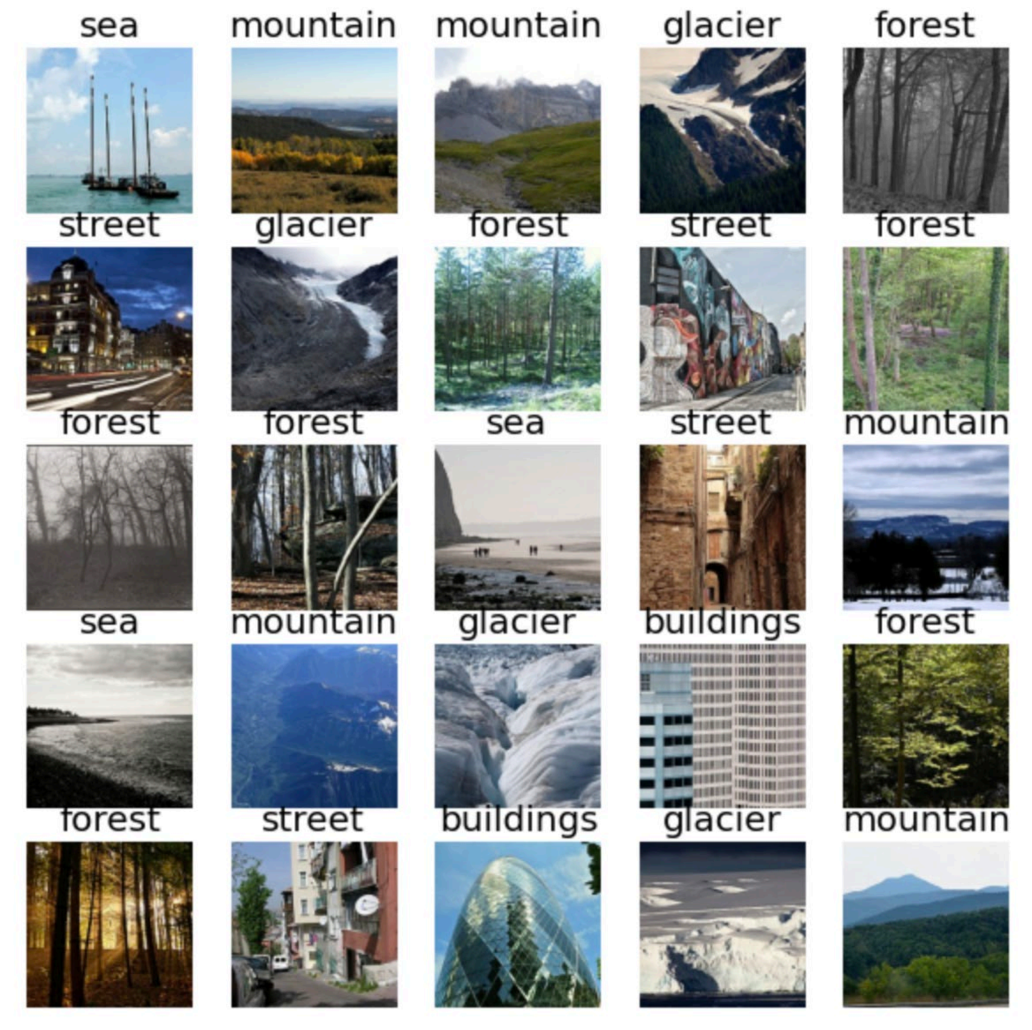


Figure: Sample Images from the Training Data

Preprocessing

- Unrelated images found in each class directory:
  - ➢ The Glacier class was the biggest culprit
  - ➢ Other classes had fewer irrelevant images
- Cleaned up bad data:
  - ➢ Problem images were removed from the training data

Figure A: images found in glacier's class directory.

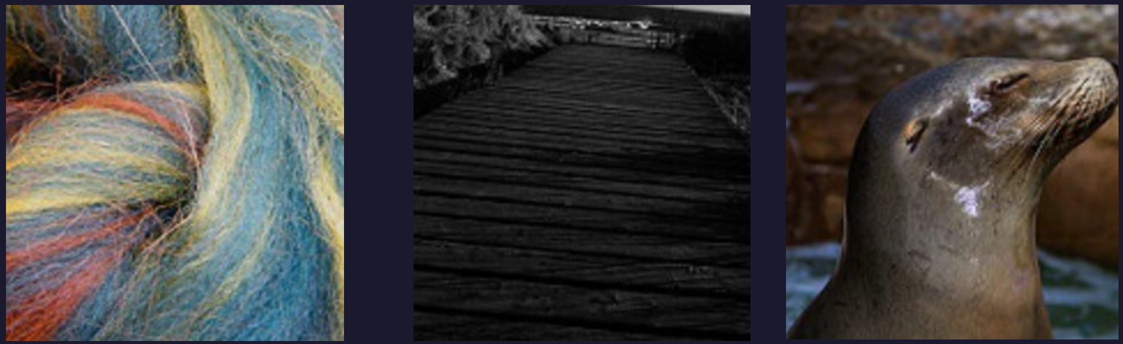


Figure B: Images found in other directories.

Train versus Test Data Points

- tf.keras.utils.image_dataset_from_directory
- Combined training and validation data in 80:20 split

Total Number of Images	16,942
• Training	11,154
• Validation	2,788
• Test	3,000

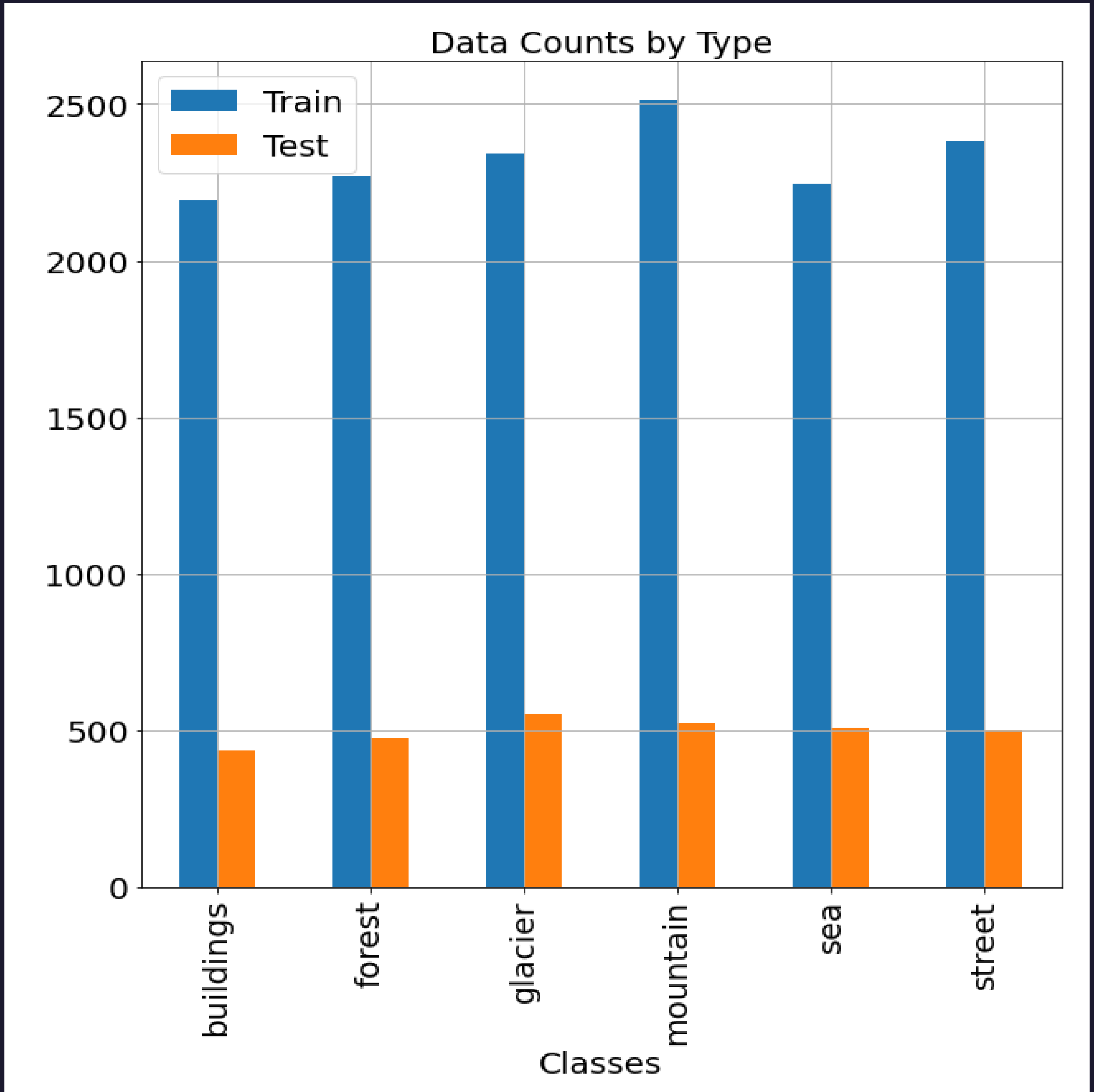


Figure: Data counts by Type

COMMON PARAMETERS

- CNN Transfer Learning using ResNet50, VGG16 and AlexNet
- Epochs: 5
- Callbacks
  - Early stopping patience of 2
  - Reduce learning by 0.1 with validation loss does not improve within 1 epoch

Model	Pretrained CNNs	Hyperparameter Tuning
CNN Case Study (Baseline)	ResNet50	Adam optimizer (learning rate = 0.001)
Model 1	ResNet50	SGD optimizer (learning rate = 0.0001)
Model 2	VGG16	Adam optimizer (learning rate=0.0002)
Model 3	AlexNet	SGD optimizer (learning rate = 0.001)

ResNet50

ResNet

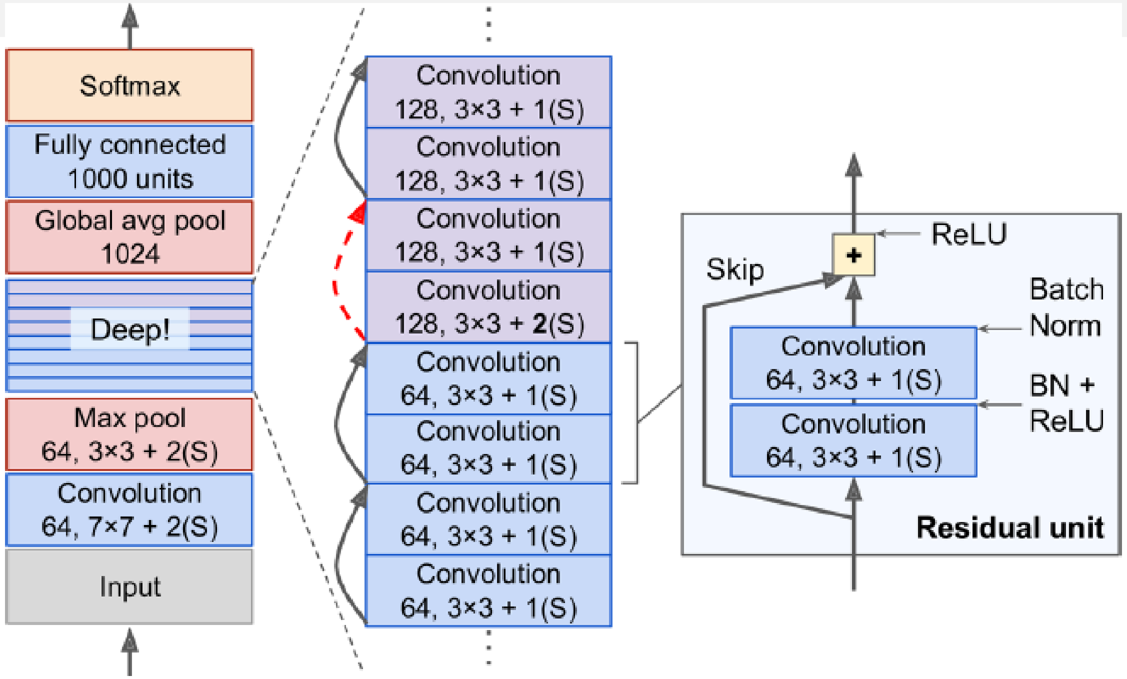
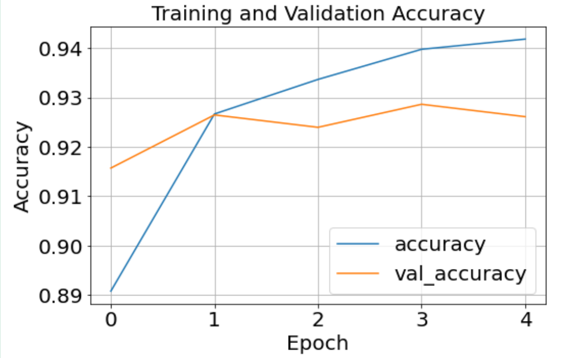
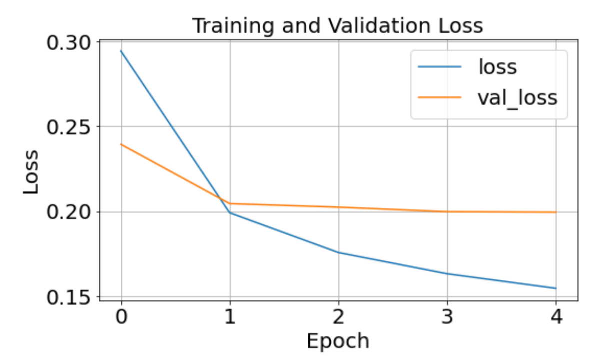
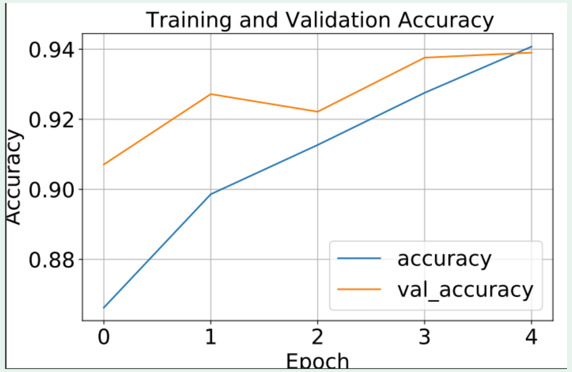
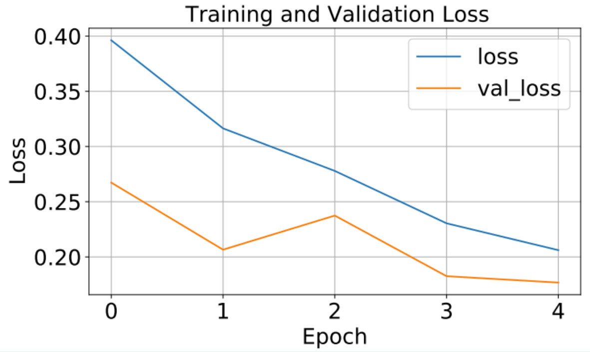
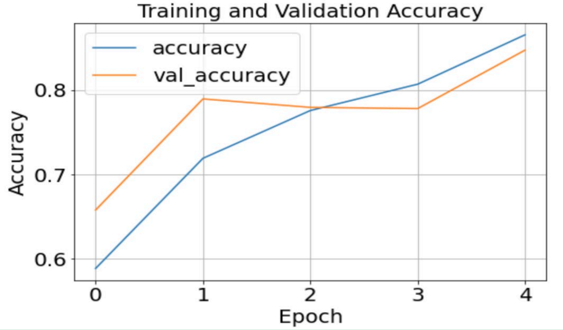


Figure 19: The architecture of ResNet. Picture courtesy of *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (2nd Edition)*.

Results

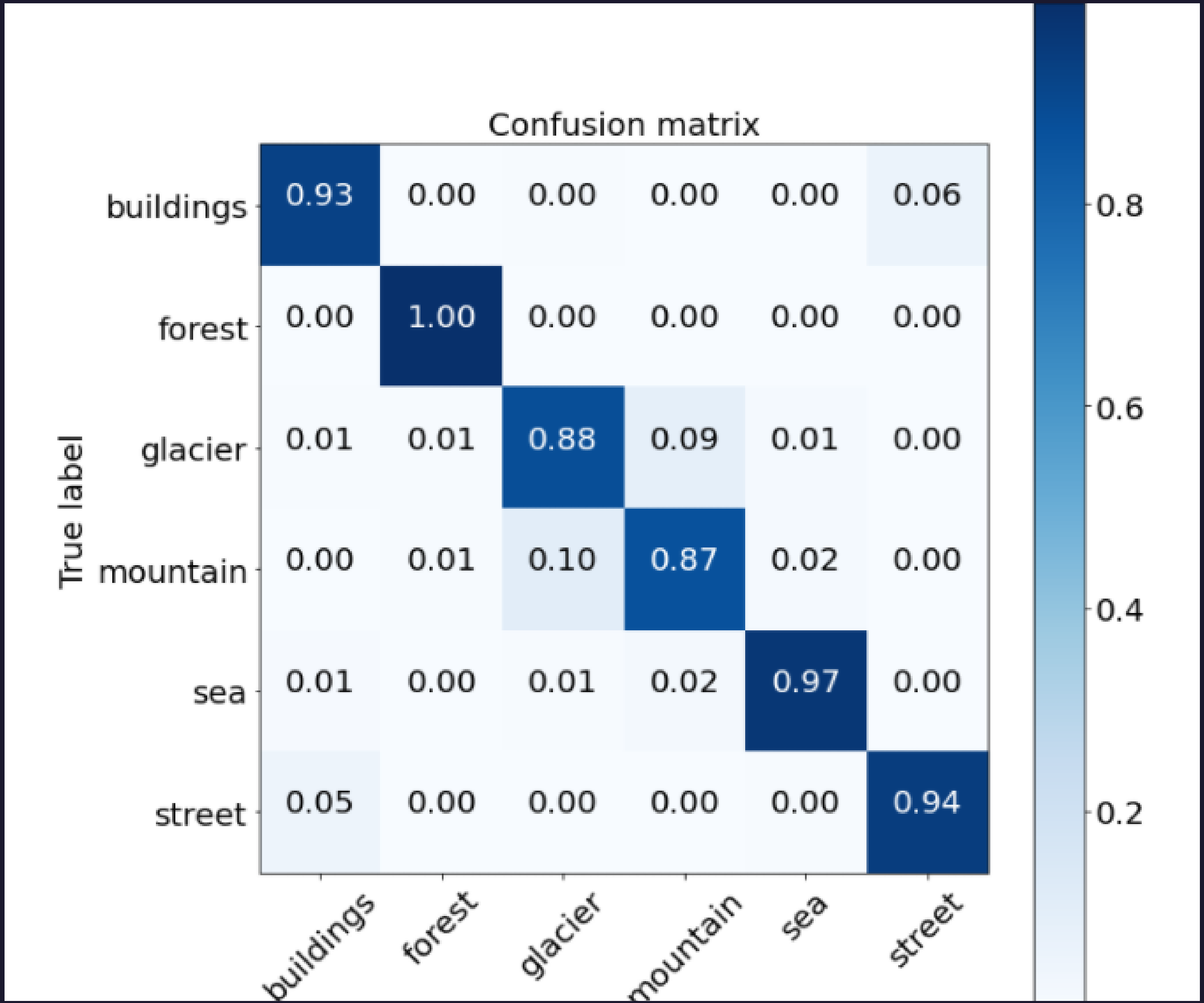
- Model 1 had the highest accuracy but only slightly higher than Model 2
- Model 2 had the lowest loss
- Collab Notebooks executed using GPU so accuracy is not reproducible

Model	**Network**	**Optimizer**	**Accuracy***	**Loss**
CNN Case Study	ResNet50	Adam optimizer (learning rate = 0.001)	0.8887	0.3414
Model 1	ResNet50	SGD optimizer (learning rate = 0.0001)	max=0.9353 μ = 0.9301 std = 0.0052 min = 0.9290	max=0.2207 μ = 0.2132 std = 0.0064 min = 0.2207
Model 2	VGG16	Adam optimizer (learning rate = 0.0002)	max=0.9310 μ = 0.9296 std = 0.0019 min = 0.9263	max=0.2089 μ = 0.2022 std = 0.0056 min = 0.1972
Model 3	AlexNet	SGD optimizer (learning rate = 0.001)	max=0.8703 μ = 0.8595 std = 0.0111 min = 0.8453	max=0.4646 μ = 0.4311 std = 0.0243 min = 0.3974

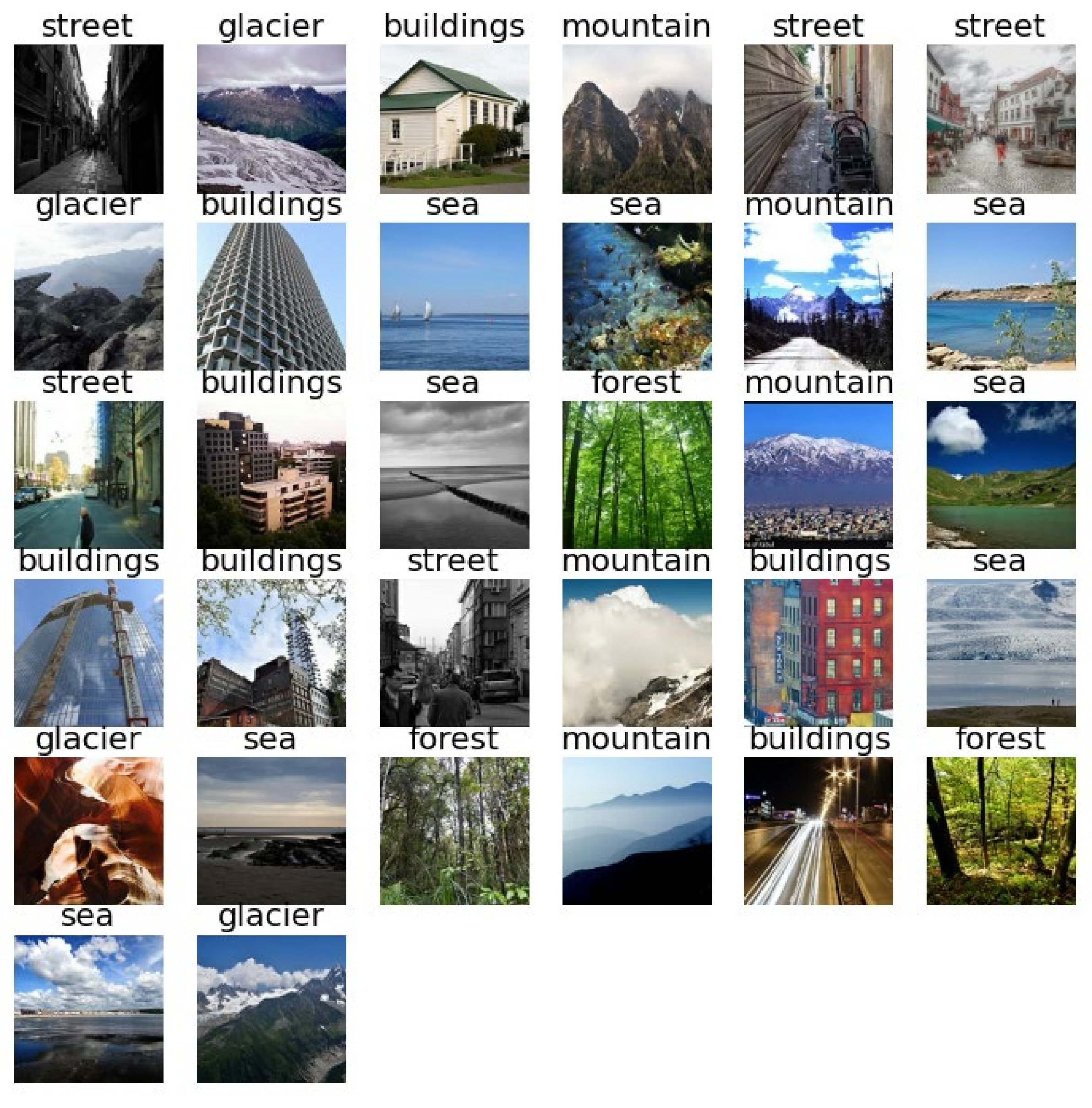
<table>
<tr><th>Model</th><th>Accuracy</th><th>Loss</th></tr>
<tr><td>• Model 1 (ResNet50)</td><td>
</td><td>
</td></tr>
<tr><td>• Model 2 (VGG16)</td><td>
</td><td>
</td></tr>
<tr><td>• Model 3 (AlexNet)</td><td>
</td><td>
</td></tr>
</table>

Normalized Confusion Matrix

- Model 1 (ResNet 50)
- Main Takeaways:
  - **Forest** class had all 474 images correctly predicted.
  - Glacier and Mountain had the smallest percentages of correct predictions.
  - Most of the incorrectly predicted images for the Street class were classified as Buildings, and vice versa.



Model 1 Predicted Images



Conclusion

- Benefits of our models
  - Create IoT drone device to identify objects in nature
  - Aircraft and ship navigation
- Power of transfer learning:
  - ResNet50 base model with accuracy of 93%
- Future work
  - Follow up on confusion matrix and error analysis

Reference

- The code for building, compiling and training CNNs were largely inspired by the following work:
  - Géron, A., 2019. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media.
- The deep utilities and pipeline (including data preprocessing, building, compiling and training DNNs) implemented in /p3_c2_s3_convolutional_neural_networks/case_study
  - Huang, Yuxiao, 2021. "Machine Learning I". Fall 2021.
- Confusion matrix function: https://github.com/imamun93/animal-image-classifications/blob/master/I_notebook.ipynb
- Load and preprocess images: https://tensorflow.google.cn/tutorials/load_data/images
- Keras Applications: https://keras.io/api/applications/
- AlexNet Architecture: https://towardsdatascience.com/implementing-alexnet-cnn-architecture-using-tensorflow-2-0-and-keras-2113e090ad98