Image Classification: CNN vs Random Forest

Shree Murthy

2024-02-19

Table of Contents

1. Problem Statement

This report aims to analyze the Intel Image Classification data using two different models: Random Forest and the VGG16 CNN model. The goal is explain why Deep Learning is required and why the Random Forest is not sufficient for this image classification task. The report will also compare the performance of the two models and discuss the results.

1.1 Why is Deep Learning Required?

To understand if deep learning was required, I created a Random Forest model. I chose this model because it is meant for classification tasks and is more complex than a simple decision tree. After running the model on the data, I found that it did extremely poorly on unseen data. Below is a detailed classification report and confusion matrix for the test set:

Accuracy for Test Data: 0.431
Classification report for Test Data:

	precision	recall	f1-score	support
buildings	0.26	0.72	0.38	437
forest	0.56	0.83	0.67	474
glacier	0.46	0.50	0.48	553
mountain	0.66	0.34	0.45	525
sea	0.40	0.07	0.13	510
street	0.77	0.18	0.29	501
accuracy			0.43	3000
macro avg	0.52	0.44	0.40	3000
weighted avg	0.52	0.43	0.40	3000

Figure 1: Random Forest - Classification Report

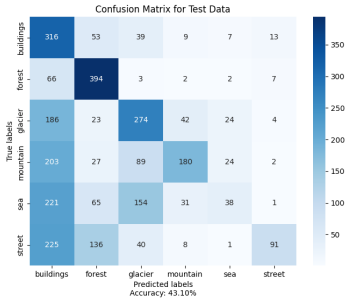

Figure 2: Random Forest - Confusion Matrix

Based on the test and train results, the model became extremely overfit. The model's training accuracy was nearly double the testing accuracy (98% and 43% respectively). Also, the classification report showed that the model wasn't able to discern the sea class. The model was heavily biased in classifying the test data as forest. Overall, the model was not able to generalize on unseen data.

Below are some example outcomes of the test predictions that further illustrate the model's inablity to classify images:

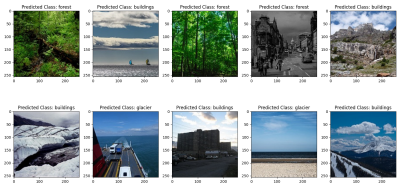

Figure 3: Random Forest - Test Predictions

Therefore, the model's poor perfomance on unseen data indicates that deep learning is more suitable for this task. This is mostly attributed to the fact that Random Forests require every pixel to be a feature, which ruins spatial relationships.

2. Exploratory Data Analysis

My dataset consists of ~12,000 images of natural scenery. The images are divided into 6 categories: buildings, forest, glacier, mountain, sea, and street. The images are of different sizes, all of them have a width of 150px but heights vary. The dataset has a seg_test, seg_train, seg_pred. Seg_pred is unlabeled; thus, it will be used for model prediction.

Analyzing the class imbalances indicated the seg_train dataset is imbalanced whereas the seg_test dataset is balanced. The seg_train dataset contains ~2k imgaes for the forest and buildings categories, the rest of the classes have less than 1k images. This could be a problem for the Random Forest model, as it may not be able to learn the features of the minority classes. The seg_test dataset is more balanced across the classes so it is a good resource for evaluating model metrics.

All the images are clear and of good quality. This indicates that I should augment them to increase adversity within the models to avoid overfitting and force them to focus on the features.

3. Data Preprocessing and Model Building

3.1 Data Preprocessing

For both models, images were scaled up to 256x256 pixels to create blurriness and increase adversity within the models. Regarding the Random Forest model, the data was flattened and the color channel was removed. For the CNN model, ImageDataGenerator was used and the

images were augmented with a horizontal flip and rotation. Images were preprocessed using the VGG16 preprocessing function. For both methods, images were preprocessed and pickled to avoid randomness when re-running the model.

Specific to the CNN model, I re-split the training data to create a train and validation set using an 80/20 split. This is done to ensure the test data is kept unseen until the evaluation step.

3.2 Model Building

The Random Forest was built using 10 estimators. A random state was applied to ensure the same outputs are recieved.

The VGG16 model was loaded using pre-trained weights (ImageNet). The top layer was removed and the layers were frozen to avoid using more epochs. Model was compiled using the Adam optimizer and categorical crossentropy loss function. I added the following layers to ensure better performance: Flatten, Dense, BatchNormalization, output layer. Model was trained with 10 epochs and employs early stopping, patience set to 3.

```
# Add custom layers
x = Flatten()(base_model.output)
x = Dense(1024, activation="relu")(x)
x = Dropout(0.5)(x)
x = BatchNormalization()(x)
x = Dense(512, activation="relu")(x)
x = BatchNormalization()(x)
x = Dense(256, activation="relu")(x)
x = BatchNormalization()(x)
output = Dense(num_classes, activation="softmax")(x)
```

4. CNN Evalaution

The VGG16 model was evaluated using the seg_train and validation data created from the preprocessing step. The model was 90%+ accurate on the train and validation set. The loss for the models were also low and the difference was within 0.1. This indicates that the model isn't too overfit. Below is the model's training loss:

Model	Train Loss	Train Accuracy	Val Loss	Val Accuracy
VGG16	0.13	0.95 (95%)	0.23	0.93 (93%)

Also, the confusion matrix, coupled with a classification report I conducted, showed that the model was able to classify the images equally well across all classes for the unseen seg_test data and unlabeled seg_pred data.

The matrix proved that it learned the features extremely well and was able to efficiently generalize. The misclassifications were minimal and can be attributed to overlapping features present in the images (ex: Image 1 and Image 6 in the Model Predictions image below).

Below is the confusion matrix of seg_test data and output of sample images from the seg_pred data:

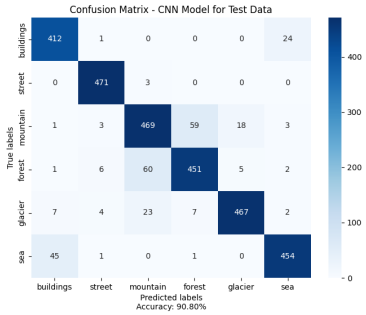

Figure 4: VGG16 - Confusion Matrix

Figure 5: VGG16 - Model Predictions

5. Discussion

In conclusion, the VGG16 model was effective in performing the image classification task and outperformed the Random Forest model. It reinforced the need for Deep Learning.

For the hyperparameters, I conducted trial and error to find the best combination. For the Random Forest I tried 100 estimators but that didn't greatly improve the accuracy of the train and test. For the VGG16 model, I tried 100 epochs with the same early stopping but the model started losing information by the time it stopped.

In the future, I would want to redo this with all the images in the dataset. For this report, I could only use ~12k images due issues encountered when trying to copy all the files from local desktop to virtual machine. The extra images could reduce class imbalances and make the models more robust. Overall, a CNN model is robust and able to take all spatial relationships

into account and generalize the features well. This was a great experience and understand when to use a Random Forest and when to use a CNN model.

6. References

- Confusion Matrix Visualization
- Image Augmentation and ImageDataGenerator
- EDA Ideas for Image Classification