Task 4 contains the material identification (classification) in which it is recognized whether the material is steel (class 0) or not (class 1). A sample picture of both is shown in Figure 1. It should be noted that Class 0 and 1 contain 78.2724% and 21.7276% of the train data, respectively.

Figure 1: Task 4: Material Classification (a) Steel (b) other materials (concrete, wood, and masonry)

Due to the size of the data set, training the whole model was not successful despite trying different methods. For instance, we tried a version of AlexNet and it was not helpful. While a reduction in train loss was observed, there is no improvement in the test loss. Additionally, the model predicted all the images as class 1. Other loss types such as focal loss and negative log-likelihood loss were also considered while no improvement is observed. However, when we used the transfer learning [x] with vgg-16 and freeze all layers except the last layer, a significant improvement was observed. For comparison, Resent30 with transfer learning was added and the model predicted well.

(10 points) What did you do exactly? How did you solve the problem? Why did you think it would be successful? Is anything new in your approach?

In essence, we used a deep convolutional neural network (CNN) to classify images. We tried different architectures from AlexNet, VGGNet, and ResNet and compared the obtained results. After gaining experience by applying available well-known architecture and evaluating their performance, we build our own architecture and applied it to the dataset. Considering the small size of the dataset, we could not achieve impressive performance, although the results look promising when data augmentation is applied [xxxxx]. Applying other methods, within the data augmentation domain, to expand the dataset will improve the results which are determined to do so after this course completion, as a future endeavor.

o (5 points) What problems did you anticipate? What problems did you encounter? Did the very first thing you tried work?

There are several challenges involved in this project. In addition to the limited size of the dataset (the train dataset has only 8312 images), we anticipated issues with imbalanced data (class 0 was 78.27% of the data; while class 1 was 21.73% of the data in the trainset).

Initially, we used the built-in structure based on AlexNet with no transfer learning. However, based on the results obtained, we ended up revising the batch size and focal loss to improve the performance.

(10 points) How did you measure success? What experiments were used? What were the results, both quantitative and qualitative? Did you succeed? Did you fail? Why? Justify your reasons with arguments supported by evidence and data. Make sure to mention any code repositories and/or resources that you used!

The main measure for performance evaluation of the employed models was the confusion matrix, and to assure that the accuracy is increasing overall. Additionally, the F1 score is also calculated for Task 4.

One of the main observations was that training a model from scratch for such small data set was very ambitious; however, using transfer learning, significant improvements have been observed in the results. The plots of class accuracies per epoch are shown in Figure 2, for comparison:

Figure 2. The plots of class accuracies per epoch

Additional 15

What was the structure of your problem? How did the structure of your model reflect the structure of your problem?

o What parts of your model had learned parameters (e.g., convolution layers) and what parts did not (e.g., post-processing classifier probabilities into decisions)?

In the transfer learning case for Resnet30, the last FC layer and the 2 convolutional layers in the last block have learned parameters successfully.

o What representations of input and output did the neural network expect? How was the data pre/post-processed?

o What was the loss function?

We started with cross-entropy loss CE. Additionally, we tried negative log-likelihood loss (NLLL) and focal loss as they are supposed to perform better with imbalanced data. The dataset was imbalanced which was addressed by oversampling and reducing batch size. The focal loss did not improve the results. Overall, NLLL and CE results are very close, but NLLL gives slightly better accuracy (0.004 & 0.005 for class 0 and 1 accuracy, respectively).

o Did the model overfit? How well did the approach generalize?

The model predictions on the unseen data are impressive so we believe there is no overfit. The final confusion matrix of the best model employed is shown in Figure 3.

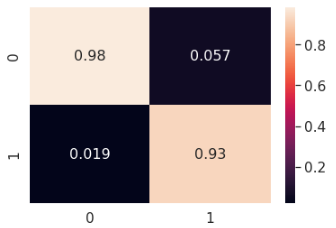


Figure 3. final confusion matrix of the best model employed

o What hyperparameters did the model have? How were they chosen? How did they affect performance? What optimizer was used?

The trial method is used to achieve an optimum learning rate. It was observed that the learning rate in the order of (1e-4) produce the most accurate results. SGD optimizer with momentum and regularization was selected. Adam is well known to perform worse than SGD for image classification tasks [xx].

The batch size was the most important hyper-parameter that affects training and results. With larger batch sizes, the model predicts only one class. As the batch size reduces the model performance improves significantly. Using a batch size of 8 the model learns in the first attempt, however, it seems like an overfit. By increasing the batch size, generalization is initiated. It is observed that most of the learning happened in the first epoch, with minor improvements afterward. While a model with a batch size of 16, learns in 10-15 epochs, its performance is lower, compared to a model with a batch size of 8. Eventually, a batch size of 8 was selected to proceed, as it yields the best model performance in terms of accuracy.

Additionally, regularization of 1e-3 and momentum of 0.9 were found to provide the best performance in terms of accuracy. Tables 1 and 2 summarizes the finalized hyper-parameters implemented in the model:

Table 1: Finalized hyper-parameter tuning results (Task 4)

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Class 0 Accuracy | Class 1 Accuracy | Accuracy |
| Built-in (AlexNet) | 1.0 | 0.0 | 0.7865 |
| VGG16 (tl) | 0.9779 | 0.9330 | 0.9663 |
| VGG16-BN (tl) | 0.9766 | 0.8900 | 0.9551 (may need more epochs) |
| ResNet30 (tl) | 0.9884 | 0.9282 | 0.9724 |

.6212 .1739 .8465 .4040 .6341

.6364 .1304 .8372 .3737 .6240

.7273 .1087 .8837 .4141 .6748

Table 2: Finalized hyper-parameter tuning results

|  |  |
| --- | --- |
| Best Model Configurations | ResNet30 (tl) |
| Batch size | 8 |
| Learning rate | 1e-4 |
| Epochs | 10 |
| Regularization | 1e-3 |
| momentum | 0.9 |
| Loss | NLLL / CE |
| Optimizer | SGD |

o What Deep Learning framework did you use?

It is noted that we have used Pytorch versions 1.13.0 and 1.12.1 on GPU.

o What existing code or models did you start with and how did these starting points help?

We used parts of the Assignment 2 code with some modifications to utilize it on the new dataset. The focal loss was attempted from the repository [xxxx], although eventually it has not been applied.

[x]  <https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html>

For image augmentation, we used learnings from:

[xxxxx] <https://towardsdatascience.com/a-comprehensive-guide-to-image-augmentation-using-pytorch-fb162f2444be>  
<https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/>

[xx]Aman Gupta, Rohan Ramanath, Jun Shi, S. Sathiya Keerthi, “Adam vs. SGD: Closing the generalization gap on image classification”, 13th Annual Workshop on Optimization for Machine Learning, 2021

[xxxx]https://github.com/gokulprasadthekkel/pytorch-multi-class-focal-loss/blob/master/focal\_loss.py