

|  |  |  |  |
| --- | --- | --- | --- |
| **Semester (Term, Year)** | Fall 2023 | | |
| **Course Code** | AER850 | | |
| **Course Section** | 1 | | |
| **Course Title** | Introduction to Machine Learning | | |
| **Course Instructor** | Dr. Faieghi | | |
| **Submission** | Project | | |
| **Submission No.** | 2 | | |
| **Submission Due Date** | Nov 26, 2023 | | |
| **Title** | Project 2 Report | | |
| **Submission Date** | Nov 26, 2023 | | |
| **Submission by (Name):** | | | **Student ID (XXXX1234)** | **Signature** | |
| Nima Dorali-Beni | | | Xxxx63711 | N.D. | |

By signing the above you attest that you have contributed to this submission and confirm that all work you contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a ”0” on the work, and “F” in the course, or possibly more

severe penalties, as well as a Disciplinary Notice on your academic record under the Academic Integrity Policy 60 , which can be found at [www.torontomu.ca/senate/policies /](https://www.torontomu.ca/senate/policies/)

Aerospace Assignment Cover as of May 2022

GitHub Link: <https://github.com/nimadb/AER_850_Project_2.git>

This project aimed to introduce the development of a deep convolution neural network (DCNN).

Image Preprocessing: The code started with preprocessing of the images. The Keras ImageDataGenerator was used allowing for augmentation techniques such as resizing, shifting, and flipping, augmenting the diversity of images while still ensuring consistency in input size and format. Additionally, normalization techniques were implemented to standardize pixel values, enhancing convergence speed during model training, and promoting stability in learning across varying features.

Architecture Design: The architecture design phase focused on constructing a Convolutional Neural Network (CNN) within a Sequential model structure. This design encompassed convolutional layers, max-pooling, and dense layers, incorporating various activation functions such as ReLU and ELU for non-linearity. The inclusion of dropout layers aimed at mitigating overfitting by randomly deactivating neurons during training. Each of these layers had a set of accompanying set of hyperparameters.

Next, the architecture and hyperparameters were adjusted to attain better results based on the loss, accuracy, validation\_loss, and validation\_accuracy. The results of the architectures, result parameters, and figures showing these results are shown in the appendix of this report.

Results: Figures 6-9 in Appendix 2 show the results of each of the models on the two testing images to be predicted. Based on the results, it can be seen that all the models can accurately predict the large crack, but have a harder time predicting the medium crack. Personally, I even found it difficult to differentiate the medium, small, and none categories of the crack images. The medium and small images just seemed to not have any crack. The large was easy to identify. That is perhaps why the models were almost a 50-50 on the medium vs none.

Overall, this was the best model:

Conv2D 32 (3,3), relu –> Conv2D 64 (3,3), relu –> Conv2D 128 (3,3), relu–> maxPooling2D (2,2) –> Flatten –> Dense 128 relu –> Dense 64 –> Dorpout 0.5 –> relu 20 epochs

Training Validation results:

loss: 0.3394 - accuracy: 0.8025 - val\_loss: 0.3673 - val\_accuracy: 0.7500

Testing Results: 56% probability prediction of the medium and 99% probability prediction of the large.

Appendix 1: Architecture – Training and Validation Results

-------------------------------------------------------------------------------------------------------------------------------

Conv2D 32 (3,3), relu –> Conv2D 64 (3,3), relu –> maxPooling2D (2,2) –> Flatten –> Dense 128 relu –> Dropout 0.5 –> Dense 64 elu –> 10 epochs

loss: 0.3682 - accuracy: 0.7556 - val\_loss: 0.3478 - val\_accuracy: 0.7500

A screenshot of a computer

Description automatically generated

Figure 1: First model.

-------------------------------------------------------------------------------------------------------------------------------

Conv2D 32 (3,3), relu –> Conv2D 64 (3,3), relu –> maxPooling2D (2,2) –> Flatten –> Dense 128 relu –> Dense 64 elu –> Dropout 0.5 –> 10 epochs

loss: 0.3602 - accuracy: 0.7638 - val\_loss: 0.3492 - val\_accuracy: 0.7500

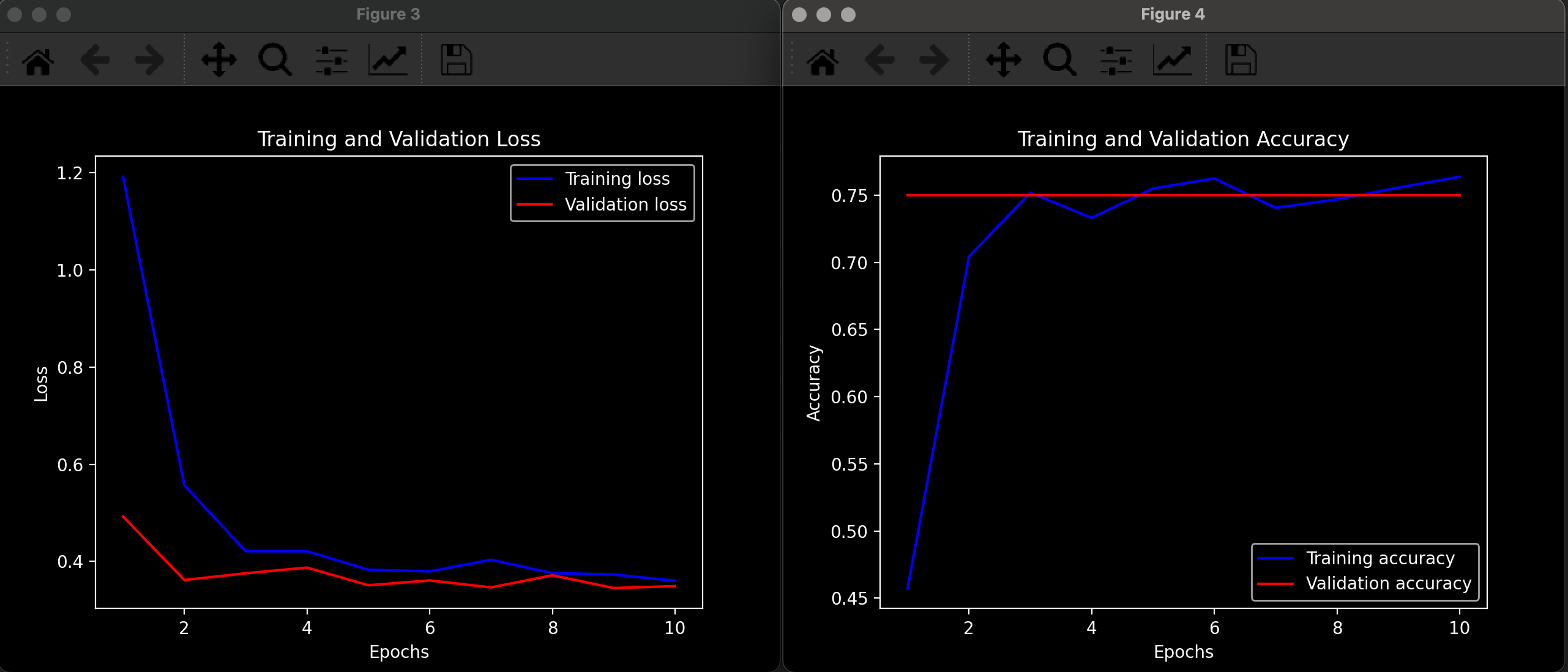


Figure 2: Second model.

-------------------------------------------------------------------------------------------------------------------------------Conv2D 32 (3,3), relu –> Conv2D 64 (3,3), relu –> maxPooling2D (2,2) –> Flatten –> Dense 128 relu –> 10 epochs

loss: 0.3714 – accuracy: 0.7681 – val\_loss: 0.4223 – val\_accuracy: 0.8413

A screenshot of a computer screen

Description automatically generated

Figure 3: Third model.

Conv2D 32 (3,3), relu –> Conv2D 64 (3,3), relu –> maxPooling2D (2,2) –> Flatten –> Dense 128 relu –> 20 epochs

loss: 0.3331 – accuracy: 0.8056 – val\_loss: 0.3429 – val\_accuracy: 0.7500

A screenshot of a computer

Description automatically generated

Figure 4: Fourth model.

-------------------------------------------------------------------------------------------------------------------------------

Conv2D 32 (3,3), relu –> Conv2D 64 (3,3), relu –> Conv2D 128 (3,3), relu–> maxPooling2D (2,2) –> Flatten –> Dense 128 relu –> Dense 64 –> Dorpout 0.5 –> relu 20 epochs

loss: 0.3394 - accuracy: 0.8025 - val\_loss: 0.3673 - val\_accuracy: 0.7500

A screenshot of a computer

Description automatically generated

Figure 5: Fifth model.

Appendix 2: Testing Results

A screenshot of a computer

Description automatically generated

Figure 6: Using model 5.

A screenshot of a computer

Description automatically generated

Figure 7: Using model 4.

A screenshot of a computer

Description automatically generated

Figure 8: Using model 2.

A screenshot of a computer

Description automatically generated

Figure 9: Using model 3.