ConvXAirBurn Report

By Ahmed and Mohit

The Convs
IIT Dharwad

23 July 2024

What we did!!

- Since this is an image classification task, we began with Convolutional Neural Networks (CNNs). We applied some preprocessing using an Image Generator, including horizontal flips, normalization, and rescaling while loading the images. Initially, we built a simple CNN model as a baseline, achieving a test accuracy of 67%, which was a good start. However, due to the class imbalance in the dataset, the model started to overfit, with a training accuracy of 87%. To address this, we added more hidden layers to extract additional features after the pooling layers, similiar to a decoder also a dropout layers to reduce training time and prioritize important weights. This improved the training accuracy to 95%, but the test accuracy only increased to 70%. The model continued to overfit, mainly due to the highly imbalanced class examples.

- To remove the overfitting was equivalent to remove the imbalance in the dataset and this could be achieved by **Data Augmentation**.

- First we Preprocessed the image then augmented it, for each image we did 5 types of augmentation, this increased our sample size 5 times.

- We build we a new CNN Encoder model with a classifier head to classify using softmax and residual block (skip connections) to help in gradient flow and also recover the past important information and on training this model on CUDA with T4 GPU for 20 epochs we got test accuracy of 90%.

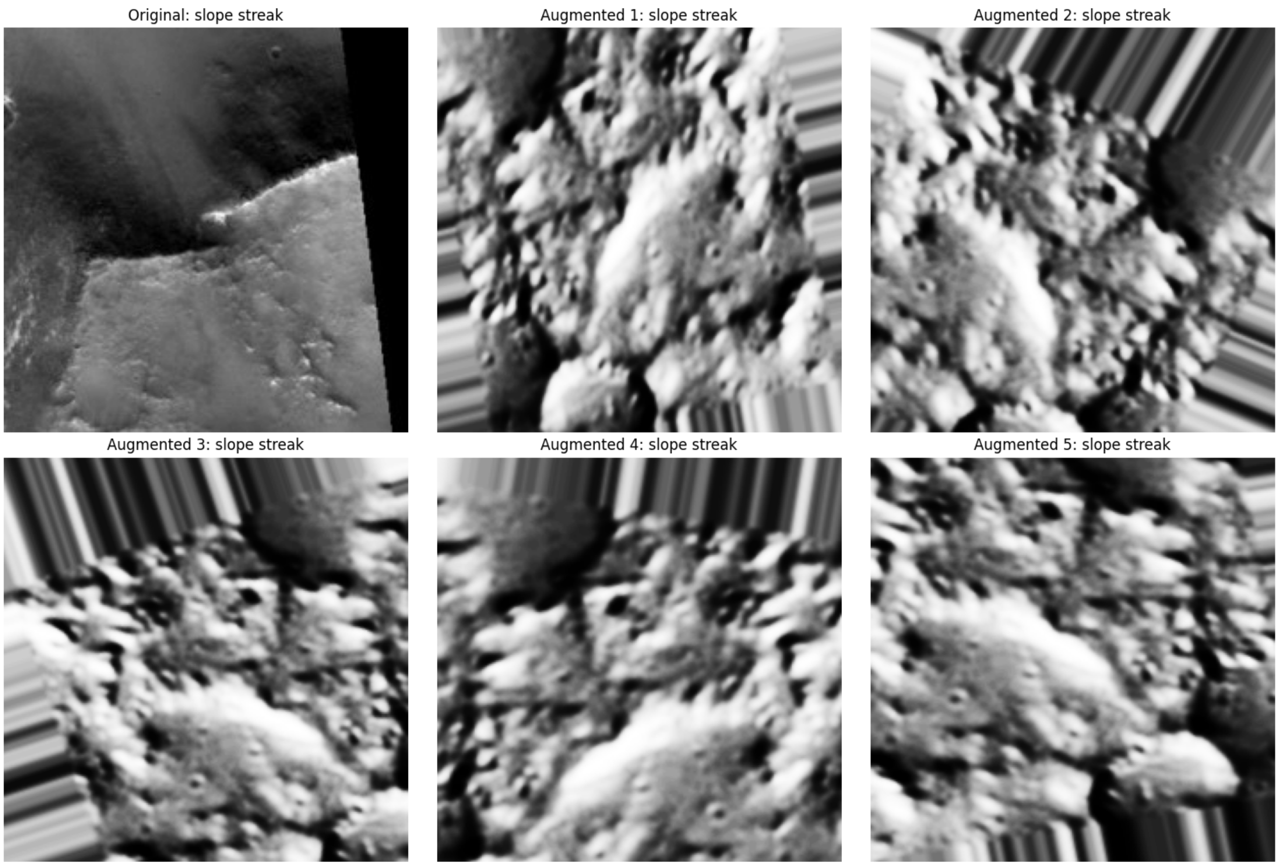
- To improve our results we made another CNN encoder decoder with classifier head and skips connections. Here we introduced a new loss which was sum of construction + prediction loss, by optimising this loss function we achieved accuracy of 92%.

- To further test on more models we thought to use Pre trained models using transfer learning, we used VGG pretrained model, unfreeze it's bottom 5 layers and without data augmentation we got good result with test accuracy of 85% and with data augmentation

Preprocessing :

- We defined the function preprocess images by reading them in grayscale, normalizing, applying Gaussian blur, and histogram equalization. Histogram equalisation enchances the important information making it easier for the model to learn.

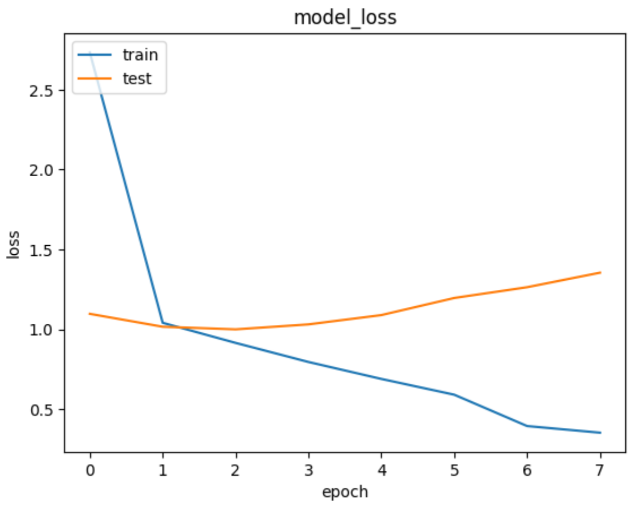
- Another function that takes each preprocessed images and augments it using different conditions like shear_zoom, horizontal flips, rotations etc. Each image has 5 corresponding augmented images. This increased our sample size.

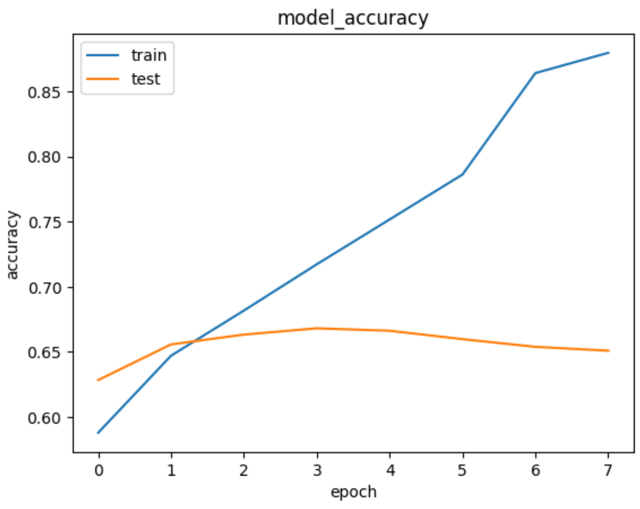

Original: slope streak
Augmented 1: slope streak
Augmented 2: slope streak
Augmented 3: slope streak
Augmented 4: slope streak
Augmented 5: slope streak

Models :

1. Simple CNN :

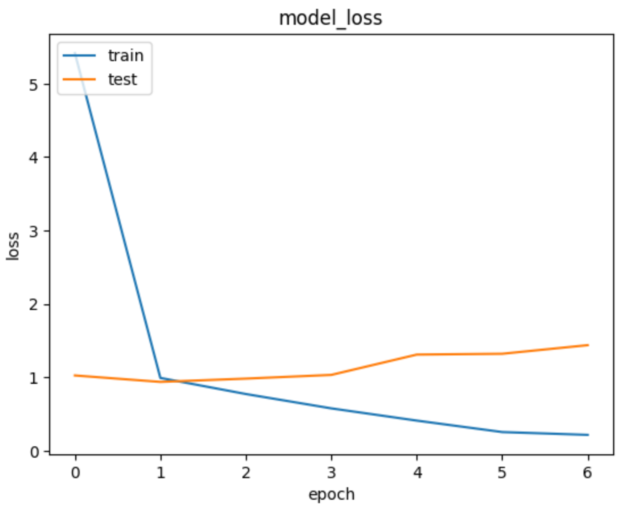
- This model contains only Conv2d and pooling layers. After training it on preprocessed images using data generator for 10 epochs we got 65% test accuracy thought the train accuracy was 87%. This model was overfitting due to imbalance in the dataset. See figure below , we can see model starts to overfit.

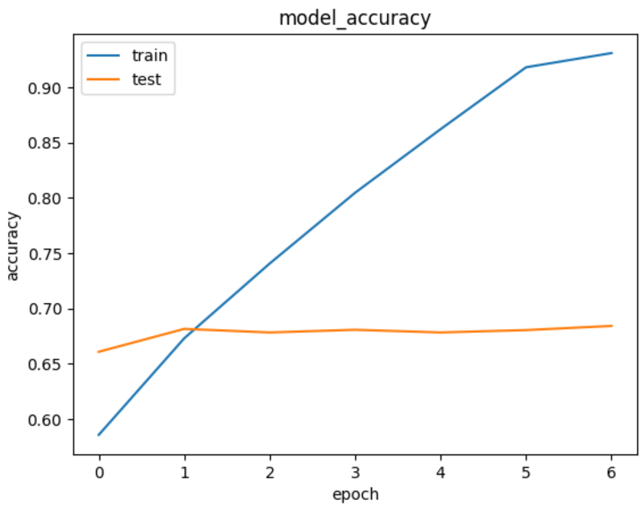




2. CNN with hidden layers :

- We added hidden neuron and dropout layers to encode more important information after cnn layers and trained it for 10 epochs this time we push our training accuracy to 95% but testing accuracy was only 70%. Our model was still overfitting.





3. CNN Encoder with Classifier Head :

- This model consists of 3 parts :

  1. **Residual Block** : Composed of two convolutional layers, each followed by batch normalization and ReLU activation, includes a skip connection that adds the input to the before final layer of residual block.

  2. **Encoder** : Begins with an initial convolutional layer, batch normalization, ReLU activation, and max pooling. It contains sequence of 3 residual blocks with increasing output channel and a stride for reducing the dimension and capturing important feautures. (downsampling)

  3. **Classifier Head** : The first layer is a **adaptive pooling layer** this layer gives equal weightage to all parameters so that important features does off and downsample the feature map to 7x7 and then flattening it and passing it to transformation layer to shape (B, class) it. (NOTE : we are using cross entropy as our loss so we don't need softmax function.)

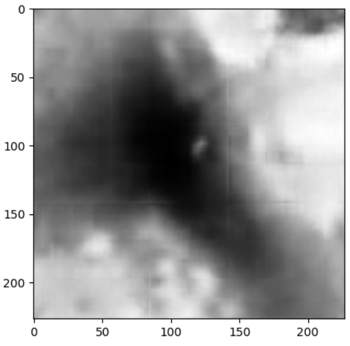
- We trained our model on the augmented data using a custom data loader in PyTorch. After 20 epochs of training, we achieved remarkable results, with a test accuracy of 90% and a training accuracy of 99.2%. Although there was some overfitting, these results were significantly better than our previous attempts and more acceptable overall.

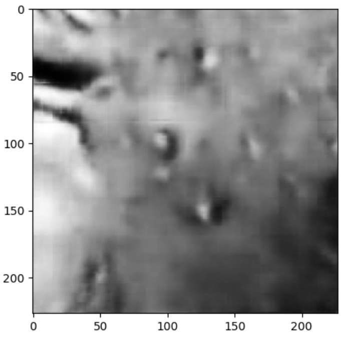
4. CNN Encoder decoder with classifier Head :

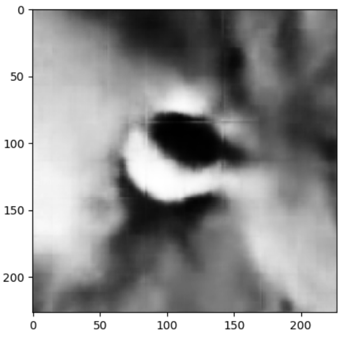
- This model architecture is same has above, but only difference is it contains a decoder part.
  - This is decoder part tries to reconstruct image given as input by using **convtranspose** , this layer performs reverse convolution or upsampling with this we also introduce a new loss function that is **reconstruction loss** so, total loss is prediction loss + reconstruction loss we try to optimise this loss function. We this training 20 epochs we push the test accuracy to 92% and training accuracy 99%.

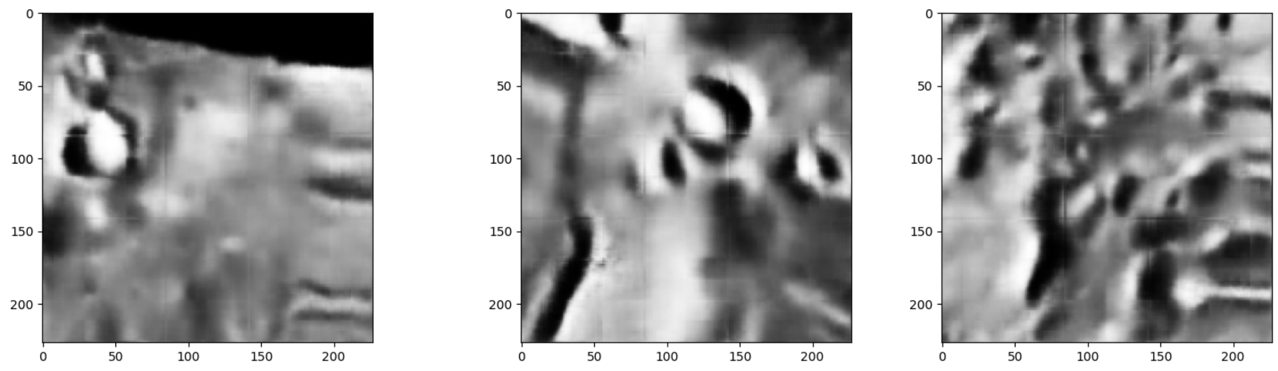
  - ****Also to ensure our data augmentation idea we trained the same model without data augmentation, training on 20 epochs we got training accuracy was 98.51% but test accuracy was only 81%.**

- Some images reconstructed using this model are given below :









5. Using VGG 16 (Transfer learning) :

- We also tried transfer learning, unfreezing below layers and trained on both augmented data and non-augmented data.
- On non-augmented data we got test accuracy of 83% for 10 epochs
- On augmented data we got test accuracy of 90% for 10 epochs.
- While training the VGG model our model started to overfit at 12th epoch and test accuracy reduced to 88%. (So training of 10 epochs was done)

ALL OUR CODES OUR AVAILBLE ON OUR REPO PLEASE VISIT IT :)).

Link to GITHUB Repo : https://github.com/MohithVelivela/SOI_SDS_2024