CSE465

Faculty Initial AzK

Group Members
Fida Ullah 2132581042
Emtiaz Mahmud Emon 2121595642

INTRODUCTION

Monocular depth estimation is an active research field due to the demand for autonomous systems to be able to understand depth, accurately, swiftly and cheaply.

In this project we explored training a model from scratch for depth estimation on the NYU Depth V2 dataset.

Code can be found using URL below
https://github.com/fidaweb/Monocular-Depth-Estimation-Training

MODEL ARCHITECTURE

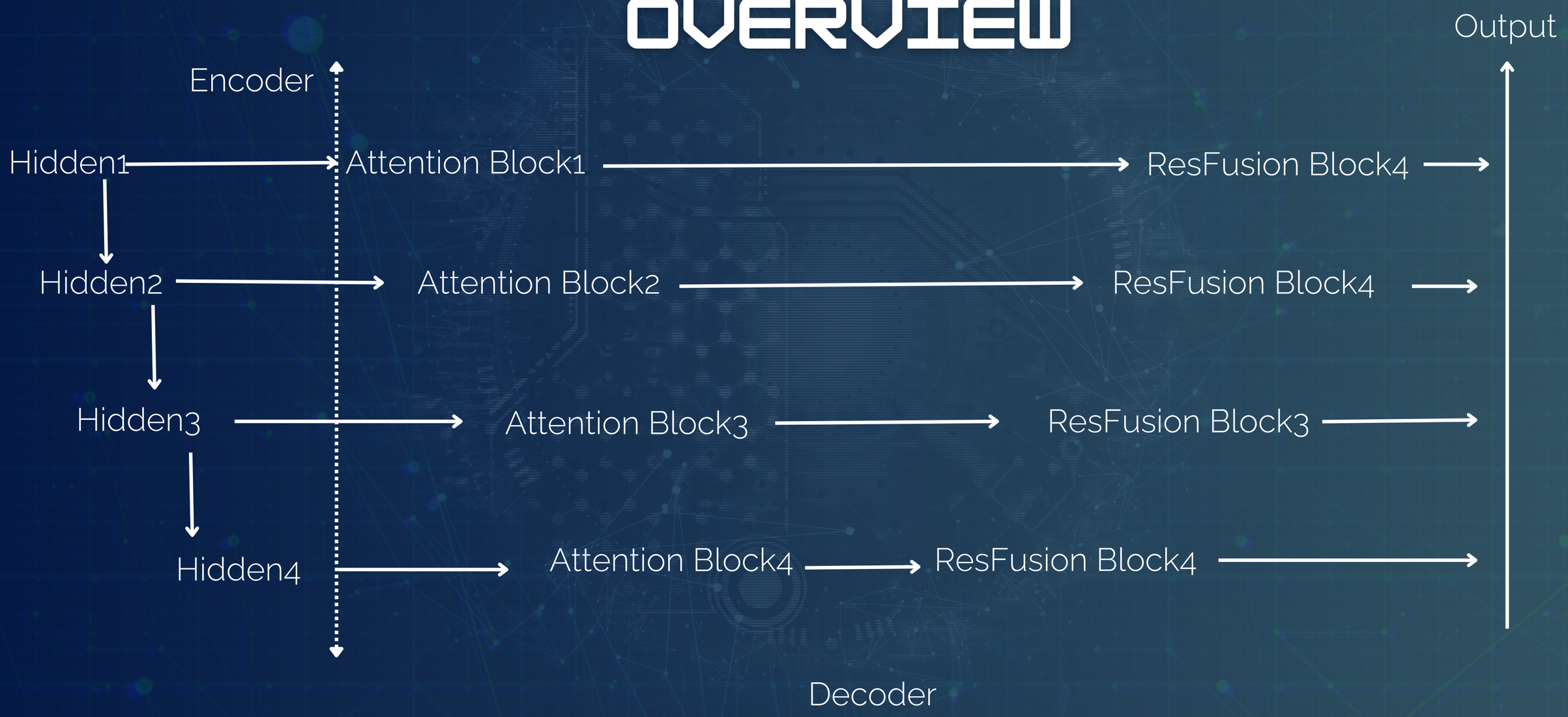
Like other models for depth estimation we adapted an encoder-decoder architecture that can decode the high level features extracted from the encoder into depth maps.

We used a pretrained encoder Dino V2 since training the encoder-decoder end to end can be costly in terms of GPU memory and be harder to train. We kept the encoder frozen using torch.no_grad().

We also tried to use the decoder class definition from DepthAnything V2 and train it from scratch without pretrained weights. However we faced the problem of vanishing gradients even with the presence of residual connections, batch normalization and LeakyReLU activations.

Thus we trained a customized model with new modifications.

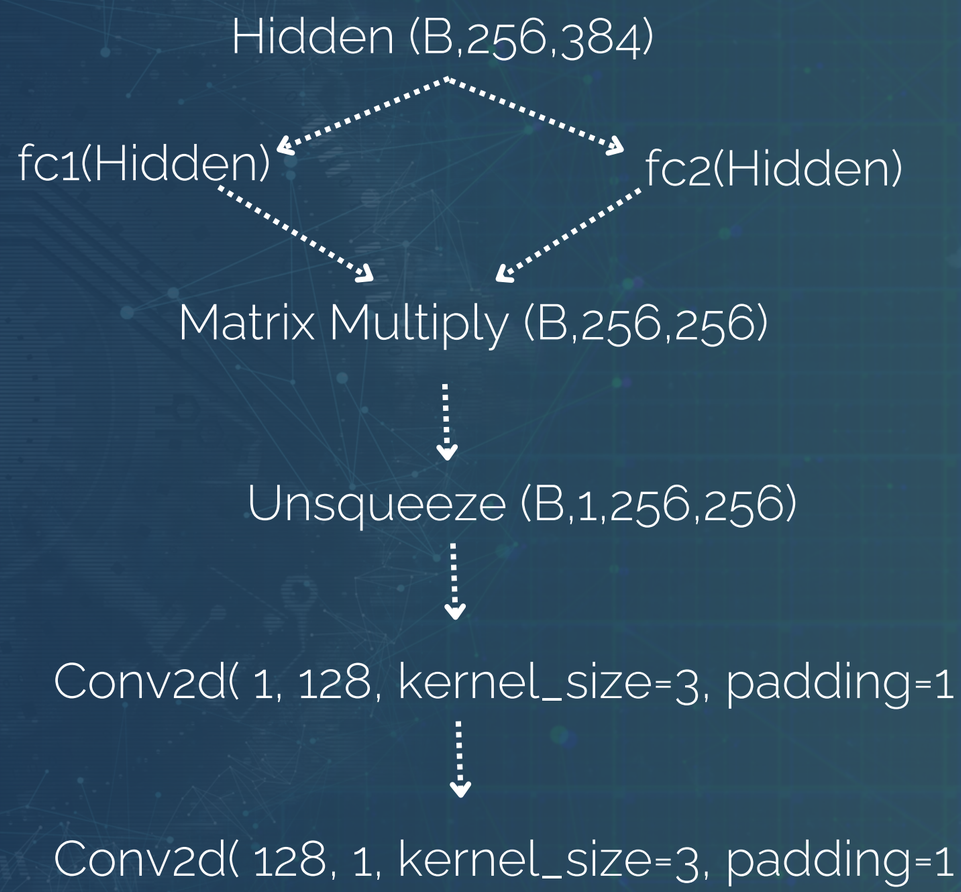
MODEL ARCHITECTURE OVERVIEW



MODEL ARCHITECTURE

ATTENTION BLOCK

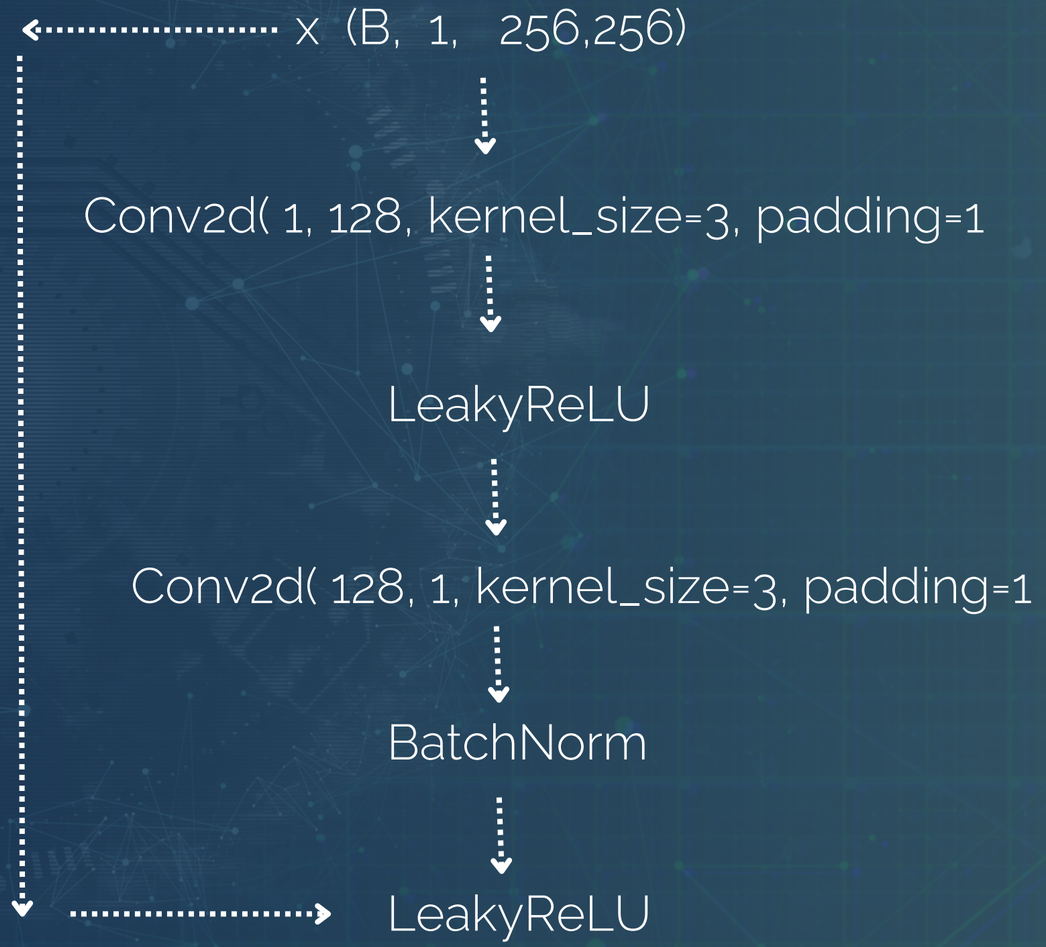
This block somewhat implements self attention on hidden states, for a global attention map on which CNN can work on. We tried to replace attention map by using only CLS embedding instead of the full hidden states but got worse performance. There is batch normalization after unsqueeze and leakyReLU.



MODEL ARCHITECTURE

RESFUSION BLOCK

This block is for further processing the hidden layers for better gradient flow.



TRAINING

We took 2000 samples out of 2000 out of around 50000 samples from NYU Depth Dataset.

We did a 60,20,20 split for training,validation and test respectively.

We used Adam optimizer with learning rate 0.005

We used Mean Squared Error as loss function for so model is penalized for fine grained details.

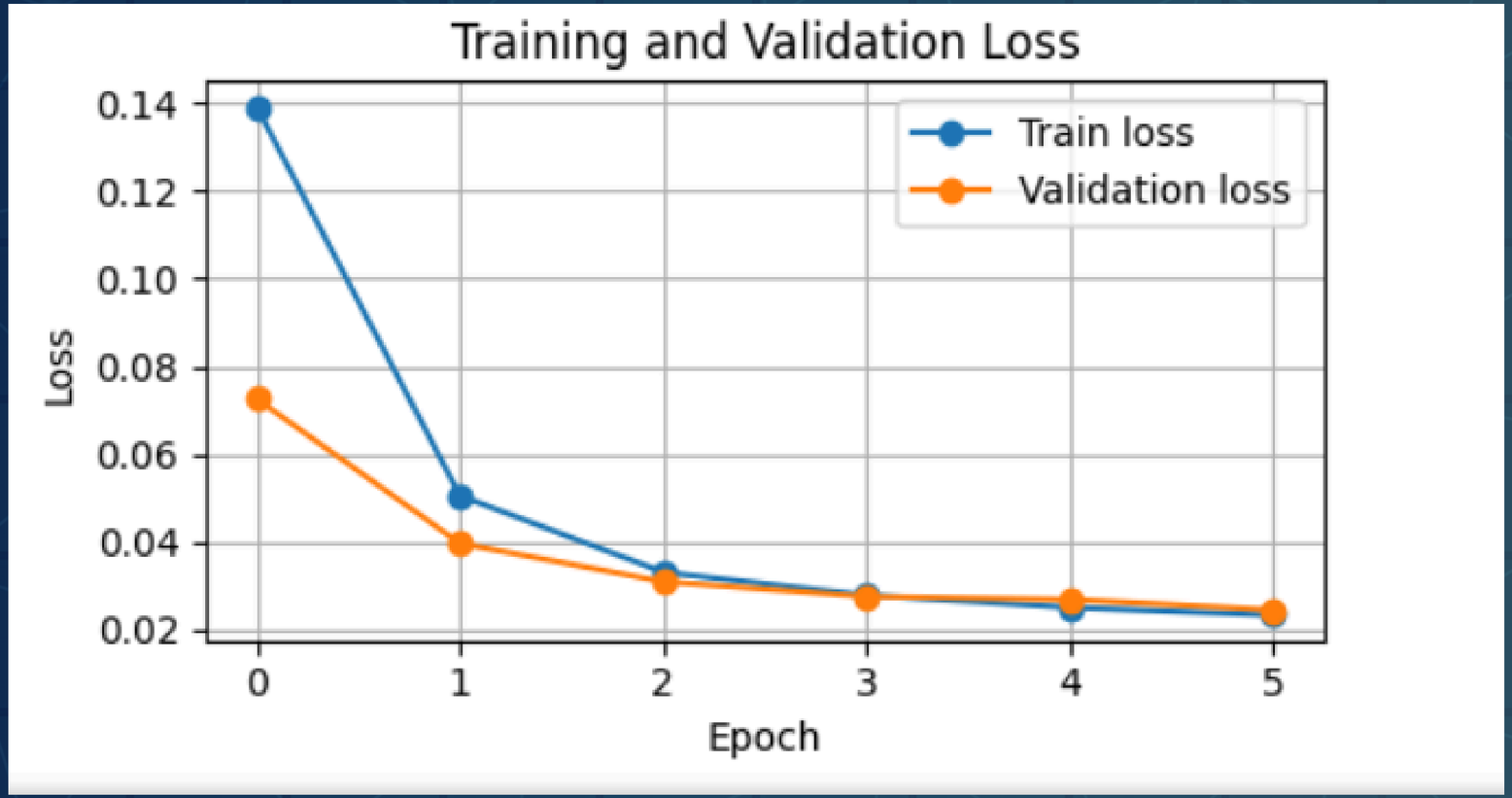
tanh(prediction)+1 was used to improve stability

We trained the model for 6 epochs and got hopeful results.

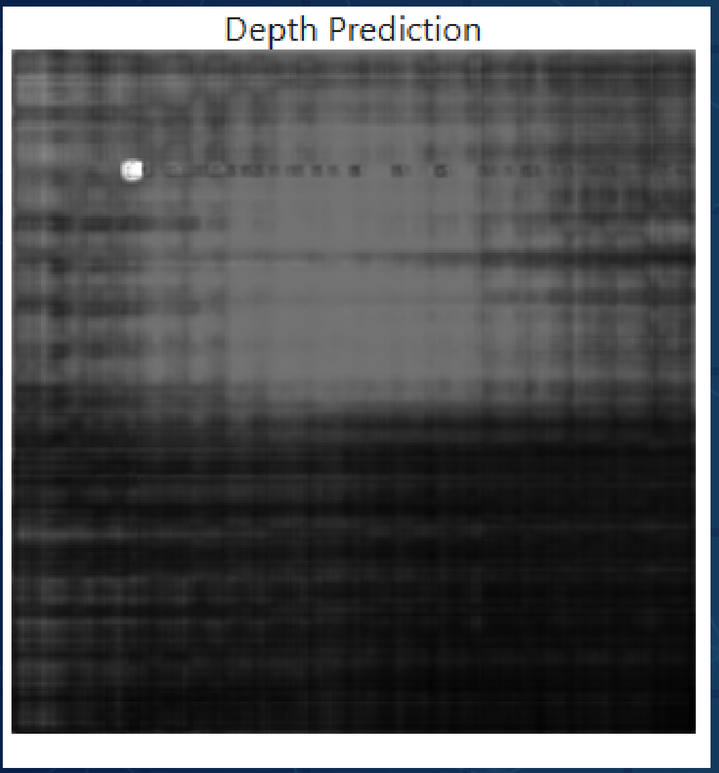
Training and Validation loss was around 0,02 for last epoch.

Test loss was around 1.149.

TRAINING


Training and Validation Loss
Train loss
Validation loss
Loss
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0
1
2
3
4
5
Epoch

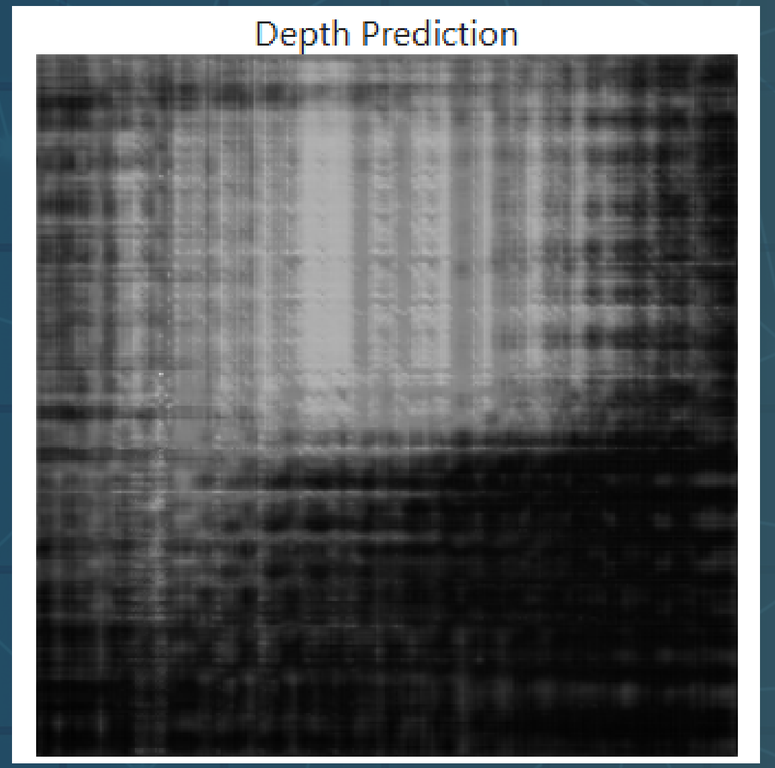
TRAINING



After training casually with few hundred samples.

Ground Truth

After training for 6 epochs with around 1500 samples from train split.




THANK YOU!