SPARSITY INVARIANT CNNs

Paper ID: 102

NNFL Design Project 2019-20

2018A8PS0415P SHIVANSH ANAND
2018A7PS0156P SHUBHECHCHHA MUDRAS
2017A7PS0113P SOOREJ S NAIR

PROBLEM

- Over the last few years, convolutional neural networks (CNNs) have impacted nearly all areas of computer vision.

- In most cases, the input to the CNN is an image or video, represented by a densely populated matrix or tensor. By combining convolutional layers with non-linearites and pooling layers, CNNs are able to learn distributed representations, extracting low-level features in the first layers, followed by successively higher-level features in subsequent layers.

- However, when the input to the network is sparse and irregular(e.g.,when only 10% of the pixels carry information), it becomes less clear how the convolution operation should be defined as for each filter location the number and placement of the inputs varies.

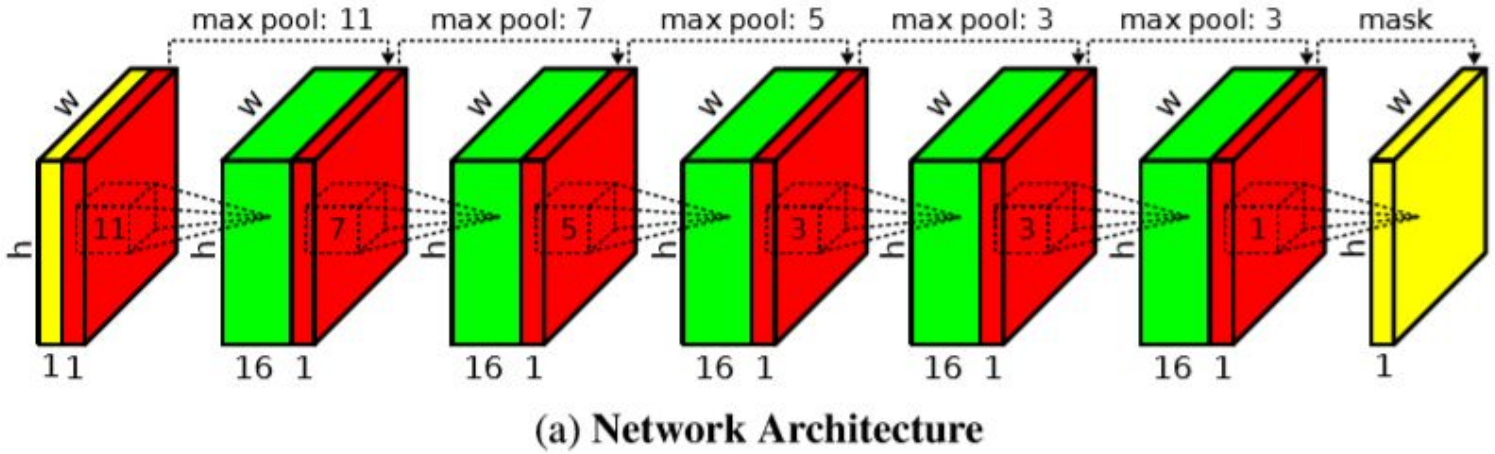
DATA

- The dataset used is a subset of the KITTI vision benchmark suite (2015).
- For developing the dataset 2 high resolution color and grayscale video cameras, a 360° Velodyne laser scanner and a GPS localization system were attached to a station wagon and it was driven around the Karlsruhe city for about 40 kms.
- The dataset we used contains over 93 thousand depth maps with corresponding raw LiDaR scans and RGB images, aligned with the raw data of the KITTI dataset.
- The images in the dataset contains less than 100% density using various methods.
- There is a ground truth(which describes the annotated depth maps) and velodyne raw (projected and temporally unrolled raw velodyne laser scans).

DATA PROCESSING

- (Optional) The image is read from the folders and is divided by 255 (multiplied by 1/255) since network trains better with data in the range of 0-1.
- This data is stored in a numpy array and is shuffled for randomness and is saved as a npy file to avoid re-execution every time program is run.
- TensorFlow requires a 4-dimensional input as contrary to 2-D data so it is converted using ***reshape*** function.
- Then a mask is created to keep track of valid inputs since convolution is only applied on valid inputs.
- Then this data is passed on to the created Sparse CNN and CNN (optional) networks.

MODEL



(a) Network Architecture

METHOD

- A novel sparse convolutional layer is introduced which weighs the elements of the convolution kernel according to the validity of the input pixels.
- Additionally, a second stream carries information about the validity of pixels to subsequent layers of the network enabling our approach to handle large levels of sparsity without significantly compromising on accuracy.
- The representation is invariant to the level of sparsity in the input.

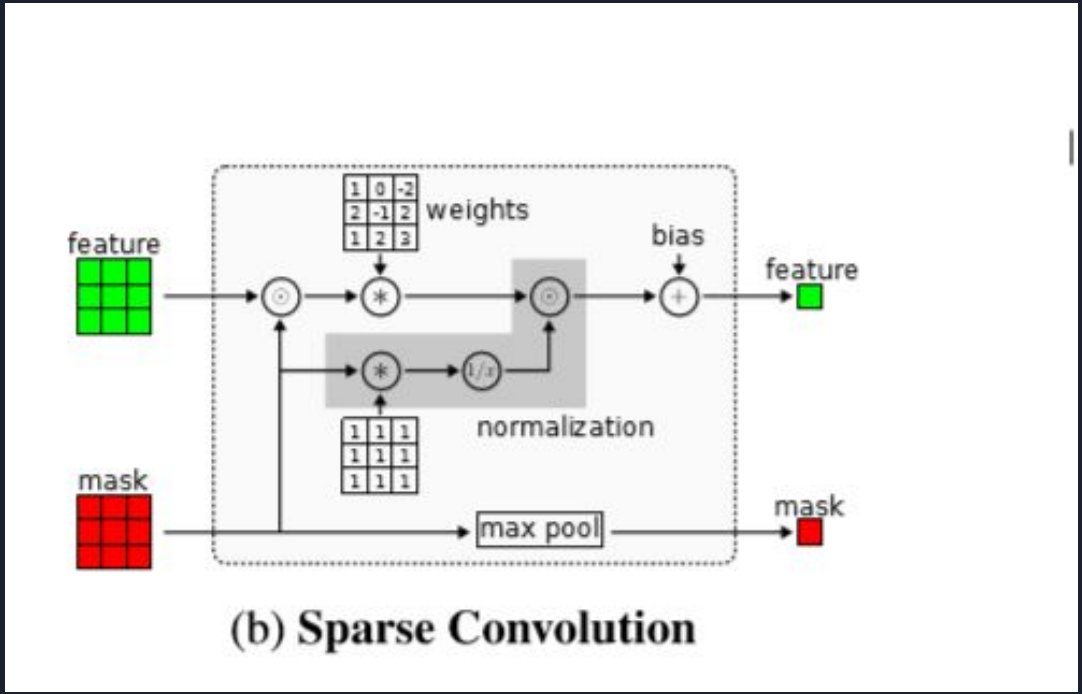
SPARSE CONVNET

- **INPUT**
  The input to our network is a sparse depth map (yellow) and a binary observation mask(red). It passes through several sparse convolution layers (dashed) with decreasing kernel sizes from 11×11 to 3×3.

- **Sparsity Invariant ConvNet**
  A Fully Convolutional Network (FCN) with five convolutional layers of kernel size 11, 7, 5, 3, and 3 are trained. Each convolution has a stride of one, 16 output channels, and is followed by a ReLU as nonlinear activation function.

- It was observed that the Sparse Convolutions converge faster than standard convolutions for most input-output combinations.

SPARSE CONVOLUTIONS



SPARSE CONVOLUTIONS

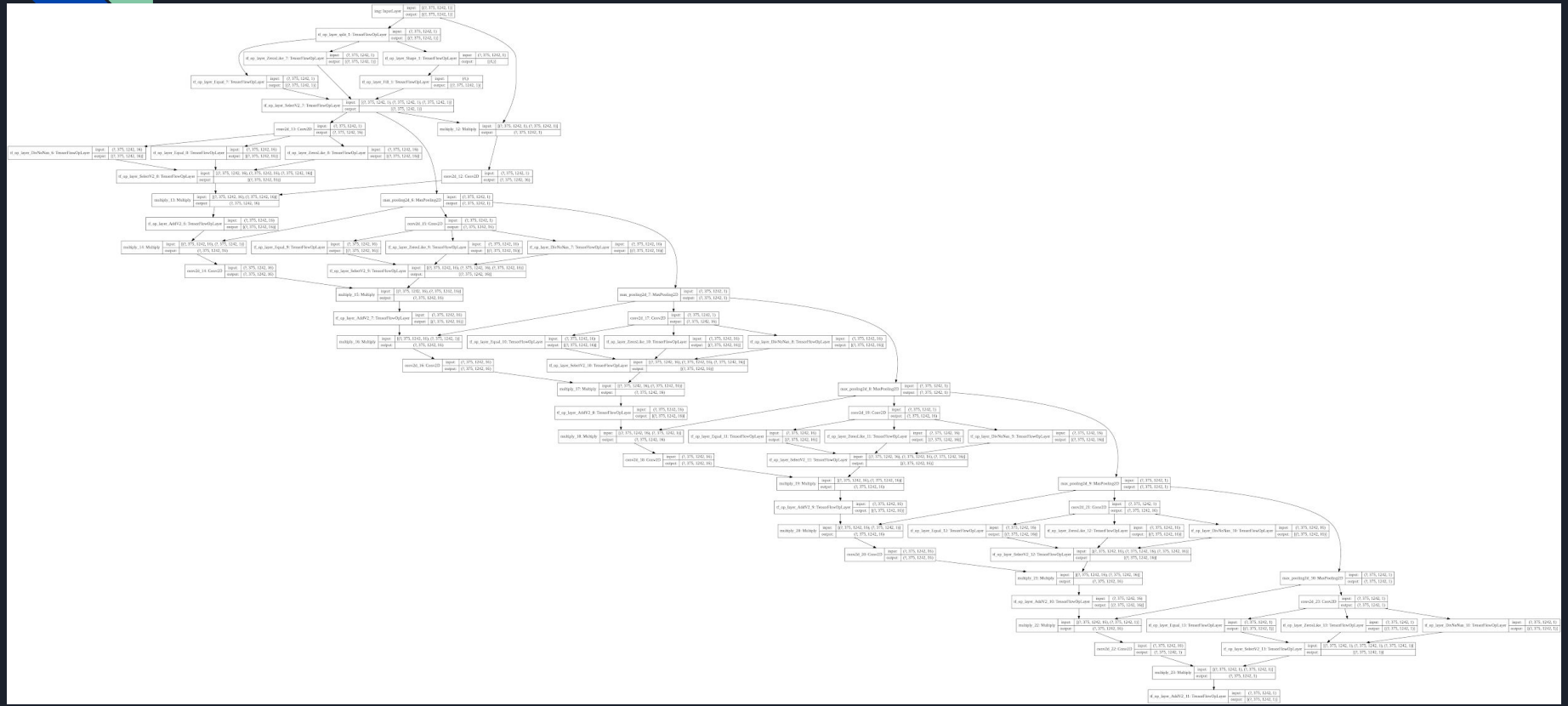
- Every Convolutional layer consists of two parts
  - The input feature which is a sparse depth map
  - A binary observation mask which carries information of valid pixels
- At every layer , the feature map goes through the following in order:
  - Element wise multiplication with the observation mask
  - Convolution operation
  - Normalisation

$$f_{u,v}(\boldsymbol{x}, \boldsymbol{o}) = \frac{\sum_{i,j=-k}^{k} o_{u+i,v+j}\, x_{u+i,v+j}\, w_{i,j}}{\sum_{i,j=-k}^{k} o_{u+i,v+j} + \epsilon} + b$$

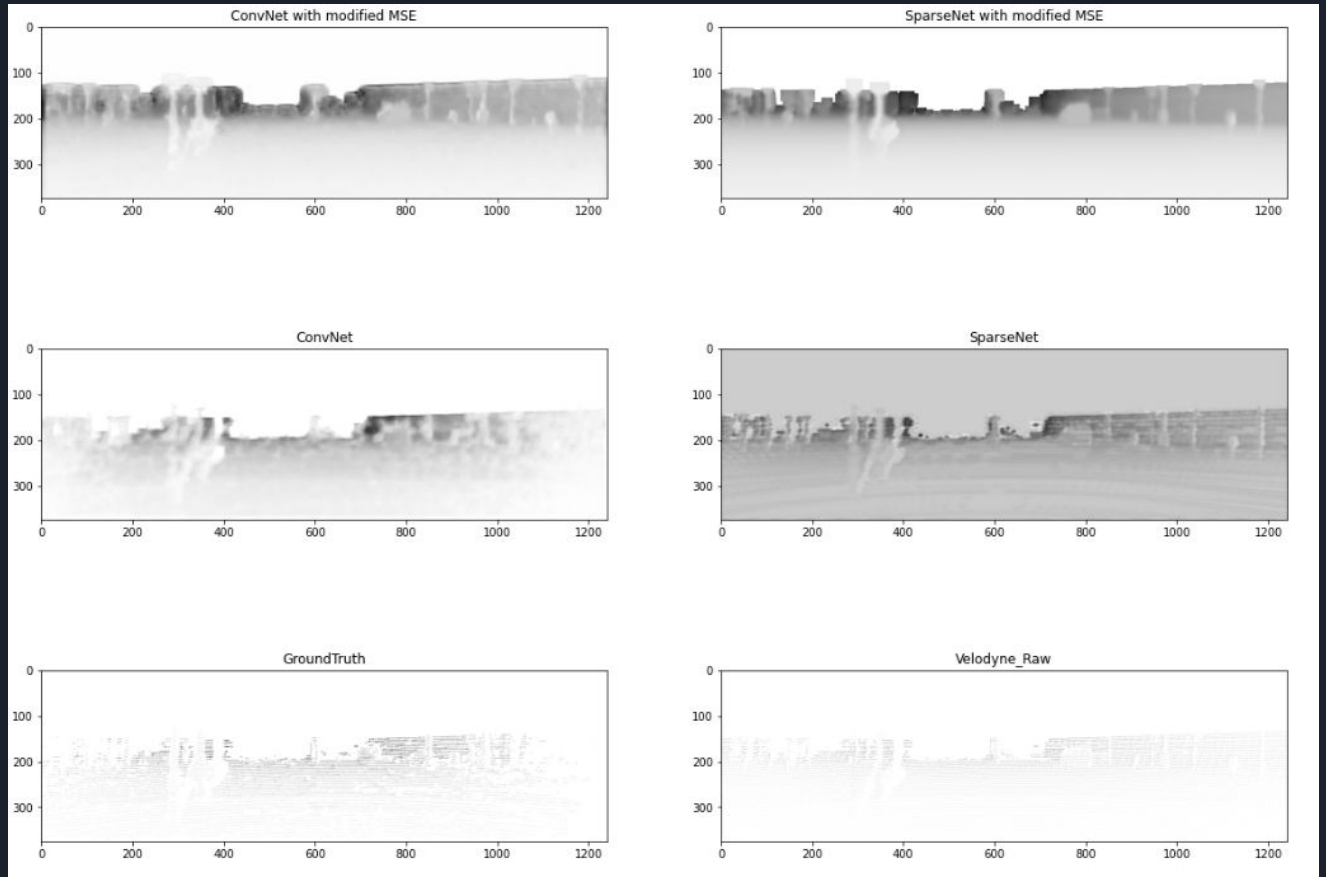
- At every layer, the mask undergoes max pooling operation

$$f^{o}_{u,v}(\boldsymbol{o}) = \max_{i,j=-k,\ldots,k} o_{u+i,v+j}$$

FLOWCHART OF THE MODEL



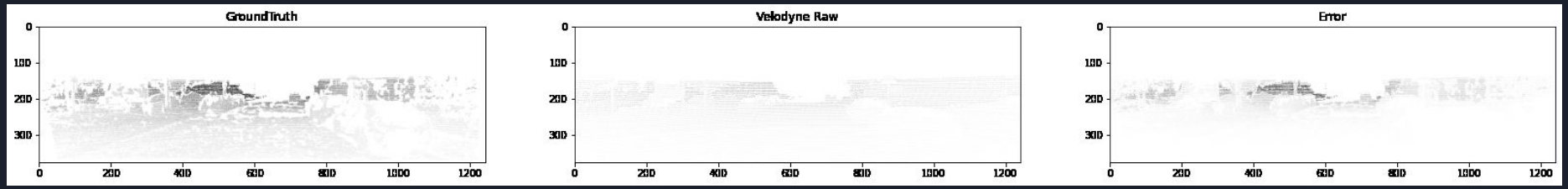
OUTPUT


ConvNet with modified MSE
SparseNet with modified MSE
ConvNet
SparseNet
GroundTruth
Velodyne_Raw
0
100
200
300
0
200
400
600
800
1000
1200

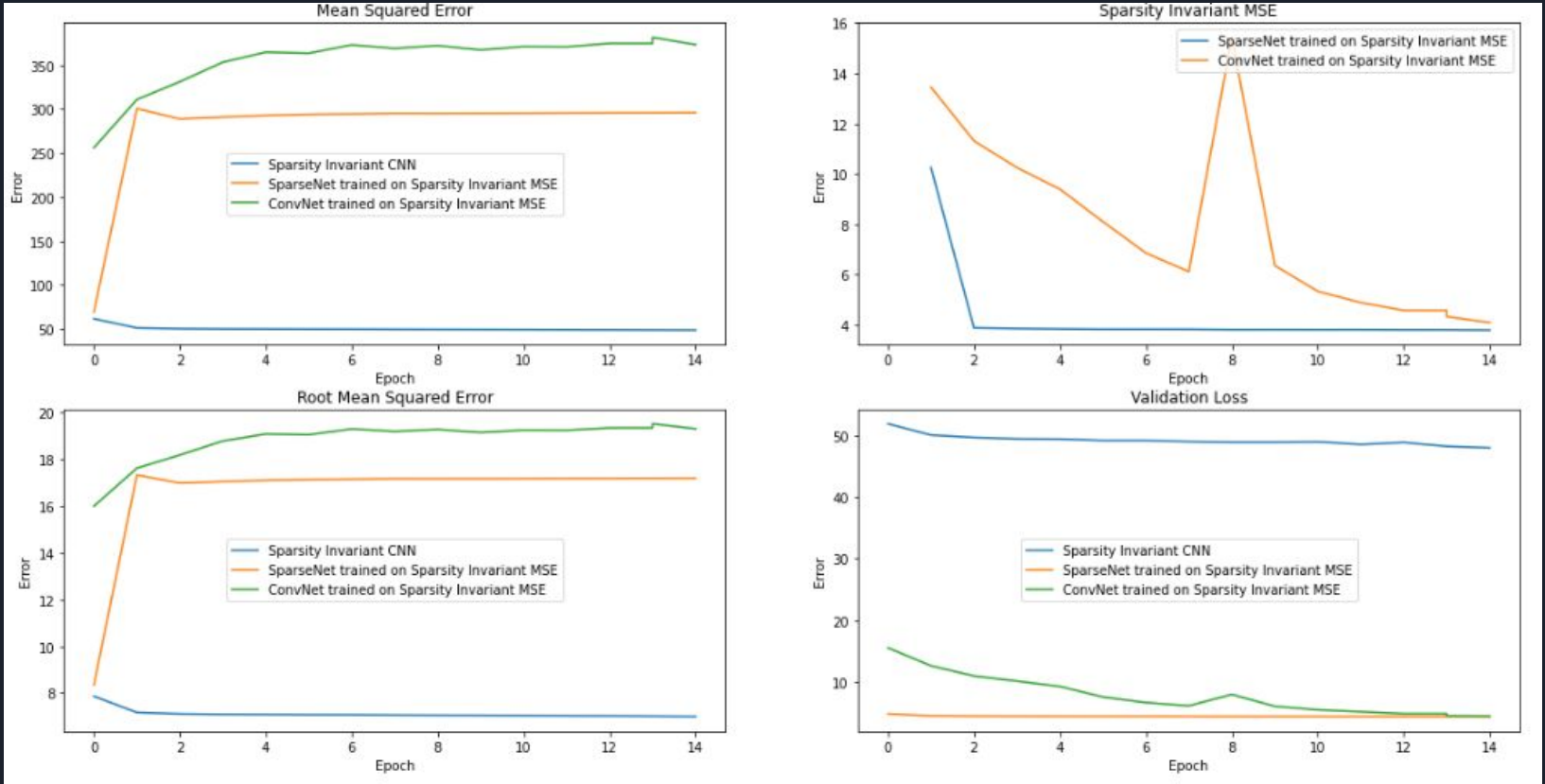
SPARSE MSE VS MSE

- From the graphs we can see that **Sparsity Invariant CNNs** perform way better than Convolution Neural Networks. It converges faster than CNNs even after training on a subset of data and produces much better output.
- It is evident from the graph that training a network on one loss may not lead to reduction in other errors. Training on **Sparsity Invariant MSE** led to increase in **MSE** by a very large value and vice versa as it was shown in the above listed table.
- The losses converged must faster in case of **Sparsity Invariant MSE** as compared to **MSE**. The reason could be the modified MSE suited the problem statement better because it is rare for groundtruth to have a 0% sparsity. This forced the network to only compare on the regions where the values are not sparse.

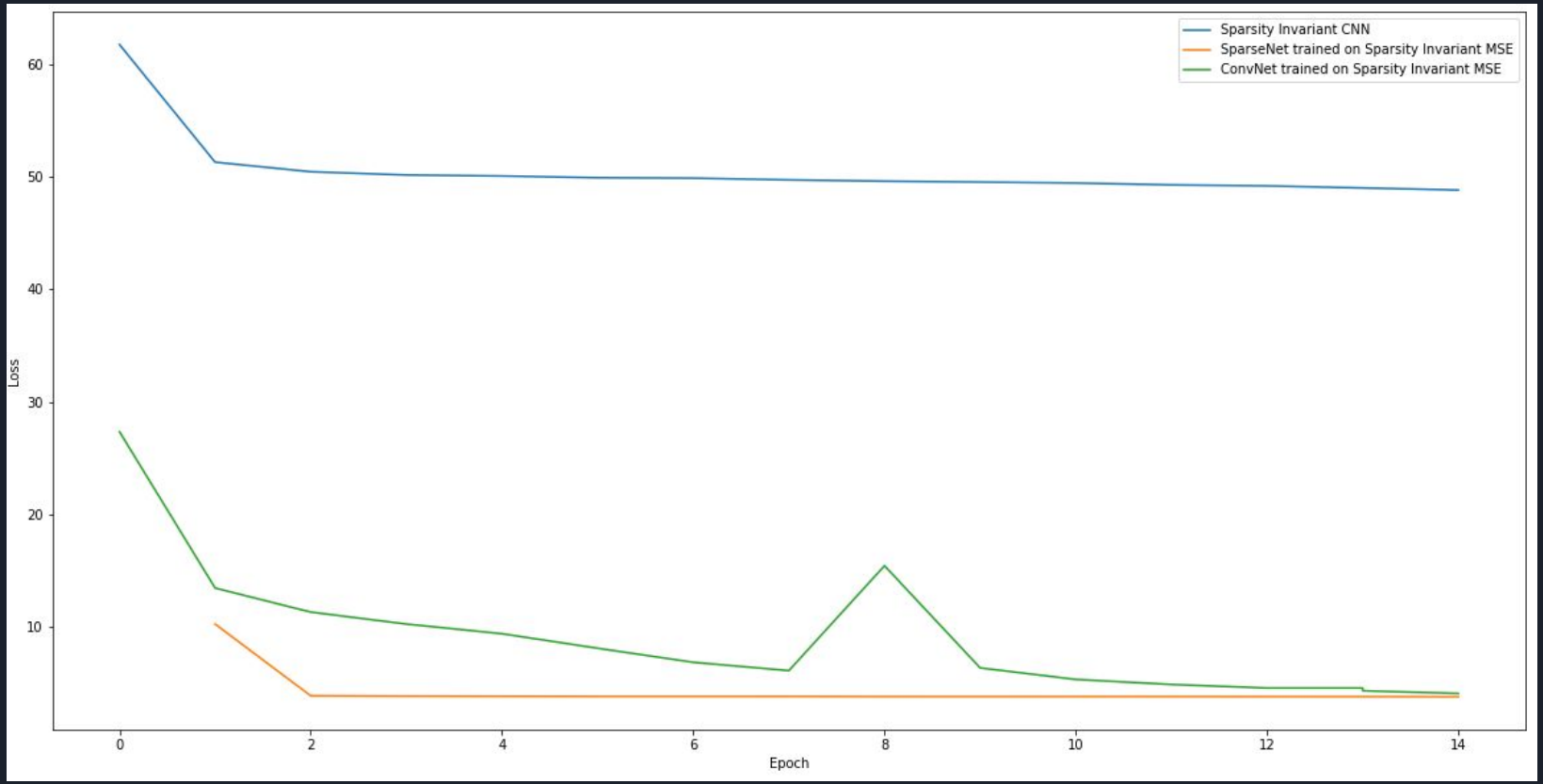
GROUND TRUTH, VELODYNE RAW AND ERROR



GRAPHS: ERRORS VS EPOCHS



GRAPH: LOSS VS EPOCHS



QUALITATIVE RESULTS

- The Sparse CNN performs better than CNN is any case however the difference is mostly visible in case of **scaled down training data**. The CNN have a tendency to not fit the data well in multiple epochs. The sparse convolution operations seems to be optimised for this problem.

- Sparse CNN have a larger convergence than the standard CNN as mentioned in the paper too.

- Training of Sparse CNN was noticeably faster and therefore it shows that Sparse CNN are highly optimised for **sparsity oriented problem statements**.

TAKEAWAYS

- We learnt about a new type of Convolution Neural Network called the Sparse Convolution model for handling sparse inputs.
- Before shifting to TensorFlow we made our project on PyTorch and it helped us learn a lot about both the libraries and advantages and disadvantages of both the libraries.
- PyTorch is more modular than TensorFlow but for troubleshooting you need to dwelve deeper into the library on the other hand when it comes to troubleshooting on TensorFlow it's as easy as using a different function from pool of thousand functions they provide.

TAKEAWAYS

- In case of our project we came to a realisation that using a standard convolution won't do any justice to the problem statement because we are ignoring the major factor of sparsity in depth maps and modifying the convolution to only work on the valid values helps us solve the problem statement.

- Sometimes using standard programming techniques and analysing a problem statement at a level deeper than just collecting the dataset and labelling it helps us solve the problem statement at a significant faster rate

Additional Observations

- The network trains better when the inputs are scaled down between 0-1 it might be due to TensorFlow being better at handling smaller errors or that scaling down the inputs and outputs helps in scaling down the error and hence the gradient is manageable.

- Neural Networks are highly optimised for training on GPU the average time per iteration on CPU for training was 28 seconds for CNN and 180 seconds for Sparse CNN. On GPU it was 2.2 seconds for CNN and 1.8 seconds for sparse CNN.

References

Original Paper:

Uhrig, J., Schneider, N., Schneider, L., Franke, U., Brox, T., & Geiger, A. (2017, October). Sparsity invariant cnns. In *2017 International Conference on 3D Vision (3DV)* (pp. 11-20). IEEE.

Open Source Resources:

TensorFlow Sentdex YouTube Playlist , TensorFlow, NumPy, Pandas, Python, Google Colab, Matplotlib