Sumit Tagadiya

+91 7043105881 | tagadiyasumit@gmail.com | sumit-tagadiya | sumittagadiya | sumittagadiya.medium.com

Education

K J Institute of Engineering and Technology (GTU) Vadodara, Gujarat
Bachelors in Computer Engineering (CGPA - 8.60 Out of 10) *Aug 2016 - May 2020*

Technical Skills

Main Skills: Data Analysis, Machine Learning, Deep Learning, NLP, Computer Vision
Languages: Python, SQL (Postgres), JavaScript, HTML
Frameworks: Tensorflow, keras, Scikit-learn, Flask
Developer Tools: Linux, Git, Docker, Google Cloud Platform, VS Code, Heroku, AWS
Libraries: Pandas, NumPy, Matplotlib, SciPy, NLTK, BeautifulSoup, spaCy, Gensim

Projects

Image Super Resolution Jan 2021 – Feb 2021

| *Resnet, CNN, Tensorflow, Flask*

- Trained a Image Super Resolution model using Enhanced Deep Residual Networks for Single Image Super-Resolution **(EDSR)** and Wide Activation for Efficient and Accurate Image Super-Resolution **(WDSR)**.
- I have trained EDSR and WDSR model on **DIVerse 2K** resolution high quality images with **Bicubic** scale 4 downgrading and **Unknown** scale 4 downgrading. Different **Augmentation** Techniques have been applied on Train data because there were only 800 images for training.
- To check quality of model **PSNR**(Peak Signal to Noise Ratio) and **SSIM**(Structural Similarity Index) have been used as performance metric of the model and **Mean Absolute Error**(MAE) have been used as loss function.
- I have trained both model for 3 lack epochs and first 10 images from DIV2K validation data have been used to monitor validation performance. I have used custom training loop using **gradient tape** method to train both the models
- I have made whole project end to end for better experience. Find **deployed** version **Here**. Also Published a **Blog** for detailed explanation to this **project**

TGS Salt Identification Kaggle Challenge Dec 2020 – Jan 2021

| *Image Segmentation, CNN, Tensorflow, Flask*

- Trained different **Unet** models to identify Salted area from geographic seismic images.
- This was basically **Image segmentation** problem. In train data there were geographic seismic images and corresponding mask images to detect salted area. I have mapped mask images into **binary class** because i had to detect only salt area.
- **Binary Cross-Entropy** has been used as loss function and **IOU**(Intersection Over Union) has been used as performance metric of the model. I have got **public score** of 0.79 and **private score** of 0.81 on **kaggle** late submission
- I have made whole project end to end for better experience. Find **deployed** version of AWS **Here** and on Heroku **Here**. Find a **report** on this project **Here** In which i have discussed a detailed approach and challenges.Find whole project **Here**

Image Segmentation on Indian Traffic Data Nov 2020 - Dec 2020

| *Image Segmentation, Computer Vision, CNN Tensorflow*

- Implemented **CANET** model by referring Attention-guided Chained Context Aggregation for Semantic Segmentation **research paper** from scratch.
- Trained **Unet** Model model with **resnet34** as backbone and **imagenet** pretrained weights as encoder weights.
- For Performance measure, **IOU**(Intersection Over Union) Score has used and **Categorical focal dice-loss** has been used as loss function of the model.
- The model will identify 21 different objects from image/video.

English to Italian Machine Translation with Vanilla and Attention Mechanism Oct 2020 – Nov 2020

| *NLP, Attention Mechanism, Encoder-Decoder, RNN, LSTM, Tensorflow*

- Developed a model using Vanilla Encoder-Decoder and Attention mechanism with LSTM
- Implemented Research paper of **Bahdanau's Attention** Mechanism with three scoring functions named Dot Scoring, General Scoring, and Concat Scoring from Scratch
- To compare the results of all these Scoring functions and vanilla Encoder-Decoder **BLUE Score** has been used for performance measure.

Stack Overflow Search Engine based on Semantic meaning Sept 2020 – Oct 2020

| *NLP, Semantic similarity, Tensorflow, Flask*

- Developed a **Search Engine** on Stackoverflow Data based on the **semantic meaning** of the question. When a user searches any question then the search engine would give the most similar results based on semantic meaning rather than keyword matching.
- The main task of this project is to understand the content of what the user is trying to search for and then return the most similar results in minimum time based on the user's query using semantic similarity.
- Glove pre-trained vectors, TF-IDF, Gensim Word2Vec model, and Tensorflow Hub module have been used for sentence encoding. To similarity measure, **Cosine similarity** has used.
- **Deployed** the whole project on **Heroku** and made it end to end for a better experience.
- Published a **Blog on medium** explaining my approach to this **project**

Document(Text) Classification using CNN Aug 2020 - Sept 2020

| *Text Embeddings, CNN, Glove vectors, keras*

- Trained two Deep CNN models one on **word-level embedding** and the second on **character level embedding**.
- In this task Data set contains 20 **Newsgroups text data** so the model predicts text into 20 different classes, Also Too much Preprocessing was required in this Task.
- Pre-trained Glove Vectors with 300d has been used for text embeddings. Model built using Keras functional API.

CNN on CIFAR Dataset July 2020 - Aug 2020

| *CNN, DenseNet, Tensorflow*

- Implemented **Densenet** Model by referring Densely Connected Convolutional Networks **research paper** and trained it from scratch.
- Densenet architecture is made by 2 blocks named **Dense block** and **Transition block**. Output block contains Batch Normalization, GlobalAvgPooling followed by a Softmax layer.
- By implementing Densenet Architecture I got **90%** test accuracy.

Microsoft Malware Detection June 2020 - July 2020

| *Supervised Learning, Multi-class Classification, XGBoost*

- Implemented **Multiclass classification** Model using **XGBoost** Algorithm to classify different malware files into respected classes. There was a total of 9 Nine malware classes.
- Dataset Size was too large, Almost around **200 GB**. Out of which 50GB of data was bytes files and 150GB of data was asm files.
- The main objective of this project is to minimize **multi-class log-loss** as much as possible. I got a minimum log loss of **0.001**. To get this much minimum log loss I made some high-level features like **image feature** from byte file and asm file etc. I have also used some feature selection techniques like the **Chi-Square test** etc.