Saarthi.ai Assignment (Deep Learning)

By-Sonam Verma

- **Problem Statement**

Given some text, we need to extract all the labels such as action that needs to be taken, action to be taken on which object and location where that object is present.

- **Dataset**

The given dataset consists of a train_data.csv, valid_data.csv, having columns path of audio data, transcription of the audio data, action that needs to be taken, action to be taken on which object and location where that object is present.

- **Preprocessing**

We remove HTML tags, if present any. We also perform Decontracting, Lowering the words, Removing punctuations, numericals and wide spaces.

- **Tokenization**

✓ Create a tokenizer with a vocab size of 5000
✓ Consider max_len = 10, as max len of text as 10
✓ Use pre padding and truncating
✓ Also, save the tokenizer in a json format, which will be used in testing phase (Making sure no data leakage is happening)

- **Label Encoding**

▫ Defined a function called encoder, this function takes the labels (action, object and location) and returns encoded and binary encoded vectors respectively
▫ For, y_train_action_b and y_train_action, it is how it is encoded

```
print(y_train_action_b[0], y_train_action[0])
```

```
[0. 0. 1. 0. 0. 0.] 2
```

▫ Similarly, for object and location there will be the same kind of vectorization for train, test and validation

- **Model**

FASTTEXT model has been used. Input size for our model is (None, 30). The Embedding layer outputs a 300-dimension vector for each token and these vectors are from pre-trained model. These 300-dimension vectors are then passed to a 128-unit LSTM model, whose output is again passed to 3 different output models with activation function as SoftMax.

- **Multi-GPU Training**

✓ It was implemented using MirroredStrategy, which automatically detects number of GPU's available and this supports synchronous distributed training on multiple GPUs on one server. It creates one replica per GPU device. Each variable in the model is mirrored across all the replicas. These variables are kept in sync with each other by applying identical updates.

✓ Used callbacks like LearningRateScheduler and ReduceLROnPlateau to control the overfitting playing with learning rate

✓ Used TensorBoard to save logs and ModelCheckpoint to save model

✓ Some parameters are, Glorat normal initialization, Adam optimizer with learning rate 1e-4, loss as categorical_crossentropy, 50 epochs and batch size is 100

- **Testing**

✓ After saving Tokenizer in Json format, used it to vectorize test data

✓ Used save model to predict the output

✓ This testing file will take a test.csv file and return loss and accuracy

- **Results**

✓ Both test and train accuracy is 100

✓ No overfitting observed