Project: Lip Reading

Group 10
Piyush Onkar(160050012) Apoorva Agarwal(203050018)

Dataset

We have used MIRACL VC1 dataset taken from kaggle. We have a dataset for 10 words and 10 phrases. Each word and phrase has 10 utterances spoken by 15 speakers. Speakers include 10 females and 5 male. We have chosen words from the available dataset. Intotal for each word we have 15*10 utterances.
Words: "Begin", "Choose", "Connection", "Navigation", "Next", "Previous", "Start", "Stop", "Hello", "Web".

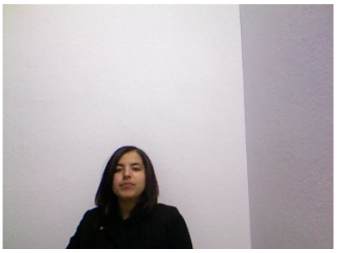
Dataset Preprocessing

1. **Crop and Align Face**
   The dataset had images of speakers with background, so we cropped and aligned faces using methods available in dlib.

   To crop face first face landmarks are marked using the shape_predictor_68_face_landmarks.dat file available with the dataset.

   To align the face we took the center of both the eyes and drew a line taking those 2 points. Using a predefined desired eye position computed the scale ratio between original image and expected image.
   Then using scale ratio and angle of eye centers computed a rotation matrix. This rotation matrix is then used to apply affine transformation over images.

   Face alignment code was there in the available code but it used some reference image. We tried to give an already aligned face as a reference image but that didn't work.
   For this part we took code from this article.



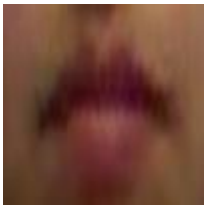
(a) Original image

(b) cropped and aligned image

2. **Crop Lips**
   We used the shape_predictor_68_face_landmarks.dat file available with the dataset for detecting face landmarks in the image.There are 68 facial landmarks detected for the face. The landmarks from number 48 to 68

correspond to the lip region. After detecting landmarks for the lip region, the lip region is cropped.

(a) Cropped and aligned image



(b)Cropped Image of the Lip

3. **Annotations**
   After the image processing we created the annotation json which had a list of image paths of each word and each for utterances.
   For this we created 2 json files, 1 file for each speaker word and respective list of all image paths in a sequence. This was used to split data according to the number of speakers.
   In another json file for each speaker, words' utterances, image paths are stored separately. This file is used to split data according to utterances.

4. **Split Train and Test Dataset**
   Dataset is split by taking 2 utterances for each word for each speaker as test data in case of a seen experiment. And it is split by taking the last 2 speakers as test data in case of unseen experiment.
   The split files had word labels for each image and normalized and resized images stored as numpy arrays.

   <u>Issue faced and how we resolved it</u>
   Issue: This process created a train file of approximate size of 5.5 GB. It was difficult to load this large into memory while training because of hardware restrictions.
   Solution: As the file had a list of image labels and image numpy we split the list into lists of maximum length 1024. Then stored the smaller lists into multiple files. This created each file of size approximately 380 MB which was much faster and easier to load.

Feature Generation

We have used pretrained vgg16, resnet18, resnet50, resnet152 models to generate different features. The feature dimensions are 12800, 512, 2048, 2048 for vgg16, resnet18, resnet50, resnet152 features respectively.

To keep the file size small we have created different feature files for different feature types.

Issue faced and how we resolved it

Issue: We planned to train vgg16 as it was done in the reference paper. But because of the large dataset and vgg16 being a deep model without gpu 1 epoch took 1hr to train.

Solution: Reference paper also used pretrained vgg16 so we generated features using pretrained vgg16 and saved them. Latter used classification models to get word labels.

Classification Model

1. BiLSTM

   This classification model we have a bidirectional lstm, a fully connected layer to get scores for each label which then is used to compute cross entropy loss and also a softmax layer which gives prediction score for labels.

   Input size to bi-lstm is the respective feature dimension.

```
class TOP_LSTM(nn.Module):
    def __init__(self):
        super().__init__()
        self.feat_dim = FEAT_DIM

        self.lstm1 = nn.LSTM(input_size = self.feat_dim, hidden_size = 1, bidirectional = True)
        self.processOut = nn.Linear(TIME_STEPS*2,N_CLASS)
        self.classifier = nn.Softmax()

    def forward(self,input):
        out,_ = self.lstm1(input.reshape(TIME_STEPS,BATCH_SIZE,self.feat_dim))
        out = out.permute(1,0,2)
        out = torch.flatten(out,1)
        scores = self.processOut(out)
        labels = self.classifier(scores).argmax(axis=1)
        return scores,labels
```

2. BiGRU

   This classification model we have a bidirectional gru, a fully connected layer to get scores for each label which then is used to compute cross entropy loss and also a softmax layer which gives prediction score for labels.

   Input size to bi-gru is the respective feature dimension.

```
class TOP_GRU(nn.Module):
    def __init__(self):
        super().__init__()
        self.feat_dim = FEAT_DIM

        self.gru = nn.GRU(input_size = self.feat_dim , hidden_size = 1 ,bidirectional = True)
        self.processOut = nn.Linear(TIME_STEPS*2,N_CLASS)
        self.classifier = nn.Softmax()

    def forward(self,input):
        out,_ = self.gru(input.reshape(TIME_STEPS,BATCH_SIZE,self.feat_dim))
        out = out.permute(1,0,2)
        out = torch.flatten(out,1)
        scores = self.processOut(out)
        labels = self.classifier(scores).argmax(axis=1)
        return scores,labels
```

Training
We have trained for 300 epochs using SGD optimizer over cross entropy loss.

Issue faced and how we resolved it
Issue : To predict label for each utterance and to decide sequence length

Solution: **LOGIC**(to predict labels for sequence of images)
As we have the dataset as features for individual frames of each utterance and we need to predict labels for complete utterance. We generate batches of size Batch_size * Sequence length. Then reshape the input to (sequence length, batch size,feature dim) before giving input to lstm or gru.
Predicted labels - By using a fully connected layer and softmax we generated 1 label for the sequence of images.
True labels - To get respective ground truth labels we took the last label from the sequence.
Choosing Sequence length
To choose sequence length we did some analysis of the dataset to get the average, max and min number of images per utterance per word for the speaker.

Word	MIN	MAX	AVG
01	5	20	10.00
02	6	19	10.04
03	4	17	11.03
04	6	22	11.95
05	5	16	9.05
06	6	20	10.69
07	7	16	9.82
08	6	18	10.49
09	6	19	11.09
10	5	16	9.12

Table1: Dataset Image Count Analysis

OVERALL AVERAGE 10.33
So we use sequence lengths of 5 and 10.

Experiments and Results

1. Seen

   The Seen Experiment means the speaker is seen by the model during training. We split the dataset by picking 2 utterances of each word for each speaker as test data.

Model	Train Accuracy	Test Accuracy	Train Loss	Test Loss
VGG16+BiLSTM	0.4456	0.2339	1.5103	2.4371
ResNet18+BiLSTM	0.3648	0.1522	1.6996	2.5188
ResNet50+BiLSTM	0.4740	0.2467	1.3776	2.3260
ResNet152+BiLSTM	0.2284	0.2339	2.0692	2.3695
VGG16+BiGRU	0.5271	0.2131	1.2279	2.8715
ResNet18+BiGRU	0.3027	0.2171	1.9644	2.4015
ResNet50+BiGRU	0.1339	0.2269	2.2110	2.3723
ResNet152+BiGRU	0.1310	0.1442	2.3004	2.3842

Table2: Model analysis for Seen Experiment

2. Unseen

   The Unseen Experiment means the speaker is not seen by the model during training. We split the dataset by picking 2 speakers as test data. Here it is obvious the test accuracy will not be good because each speaker might speak word slow or fast. This can also be observed from table2 and table3 that test accuracy is better than in the case of the Seen experiment.

Model	Train Accuracy	Test Accuracy	Train Loss	Test Loss
VGG16+BiLSTM	0.275297619	0.0875	2.054685651	3.209380388
ResNet18+BiLSTM	0.2194940476	0.10625	2.197231322	2.912849963
ResNet50+BiLSTM	0.2566964286	0.14375	2.070552934	2.894488335
ResNet152+BiLSTM	0.255952381	0.10625	2.091168744	3.129854202
VGG16+BiGRU	0.21875	0.1	2.195513833	2.997414231
ResNet18+BiGRU	0.209077381	0.13125	2.218755143	2.856699646
ResNet50+BiGRU	0.2046130952	0.10625	2.201843695	2.918291032
ResNet152+BiGRU	0.1354166667	0.075	2.303754792	3.05005151

Table3: Model analysis for Unseen Experiment

Best model is Resnet50 + BiLSTM in both Seen and Unseen settings.