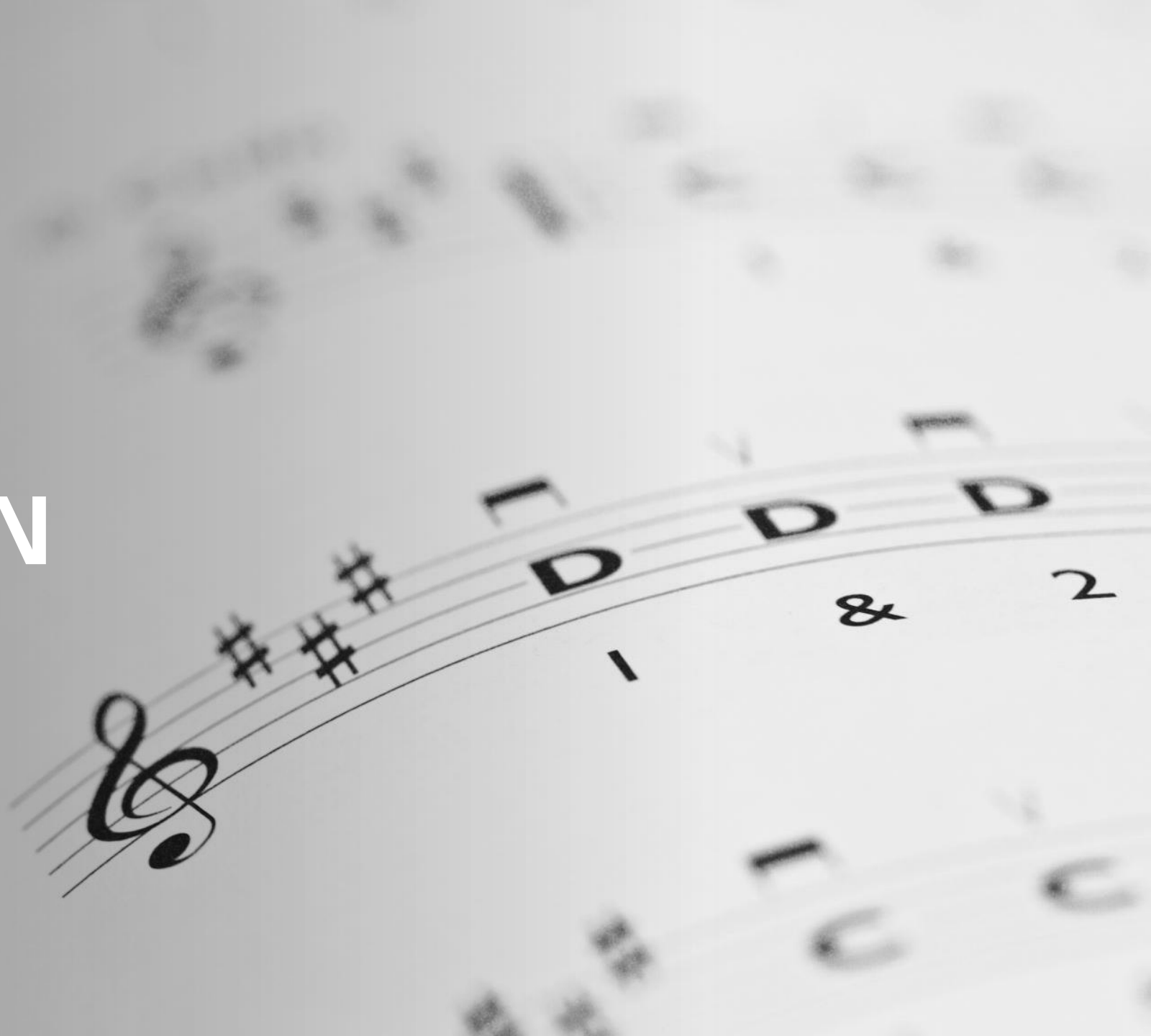
OPTICAL MUSIC RECOGNITION

- Group 16

- Instructor: Dr. Dinh Viet Sang



Members

- Vu Cong Duy - 20176737
- Dao Hong Quan - 20176850
- Tran Cong Minh - 20176825
- Tran Thi Hang - 20176748
- Duong Thi Hue - 20176772

Content

Introduction

- Reading and understanding music sheets is one of the most straightforward examples of optical tasks that can be improved with the help of computers

- Implement and improve an existing OMR neural network project to read and translate images of single-staff scores

Tasks

Vu Cong Duy:

- Implementing models
- CTC

Dao Hong Quan:

- Data preprocessing
- CRNN

Tran Cong Minh:

- Output conversion
- Optimization

Duong Thi Hue:

- Evaluation metrics
- CNN

Tran Thi Hang:

- Evaluation metrics
- RNN

Original work

- Based on Jorge Calvo-Zaragoza and David Rizo's research paper "End-to-End Optical Music Recognition of Monophonic Scores"

- Rewrote the code using Pytorch and implemented multitask learning

Variables

There are two types of input for each sample.

- The first one is a basic png file of the score.
- The other is the jpg file containing the distorted version.

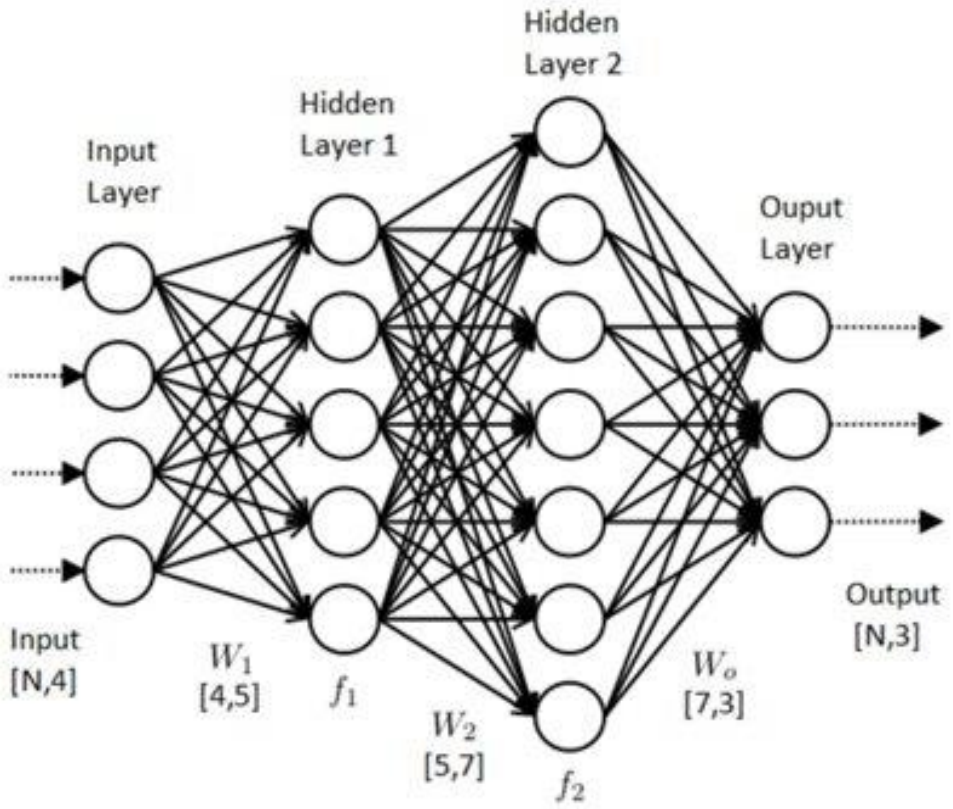
Two types of output, both are a sequence of symbols representing the image.

THEORETICAL BACKGROUND

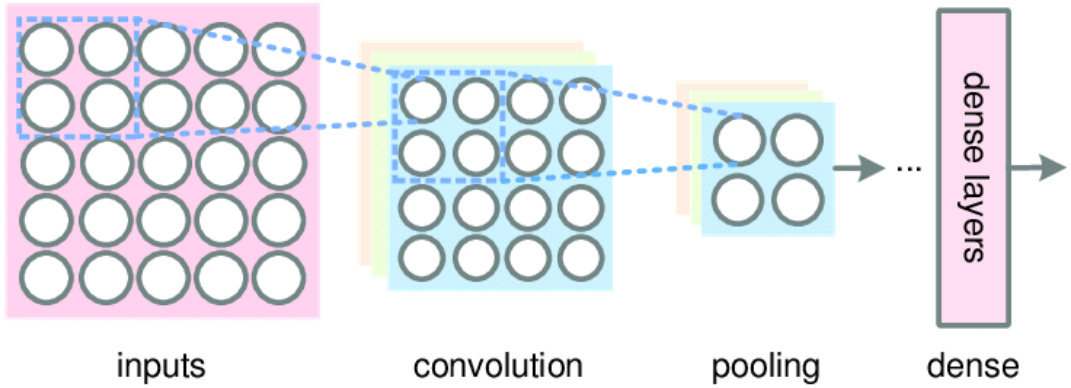
Neural Network

A traditional Artificial Neural Network:

- Input Nodes (input layer)
- Hidden nodes (hidden layer)
- Output Nodes (output layer)
- Connections and weights
- Activation function

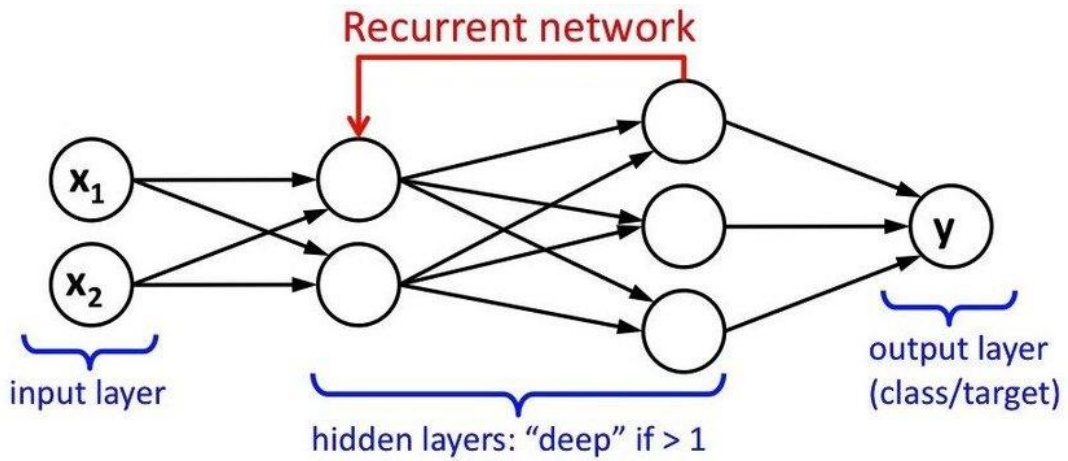


Convolutional Neural Network



- The model's number of nodes may grow exponentially with the number of layers, which leads to numerous problems
- The main difference between CNN and ANN is the convolutional layer.
- Each convolutional layer is usually accompanied by a pooling layer
- The information will be extracted for meaning and reduced in size

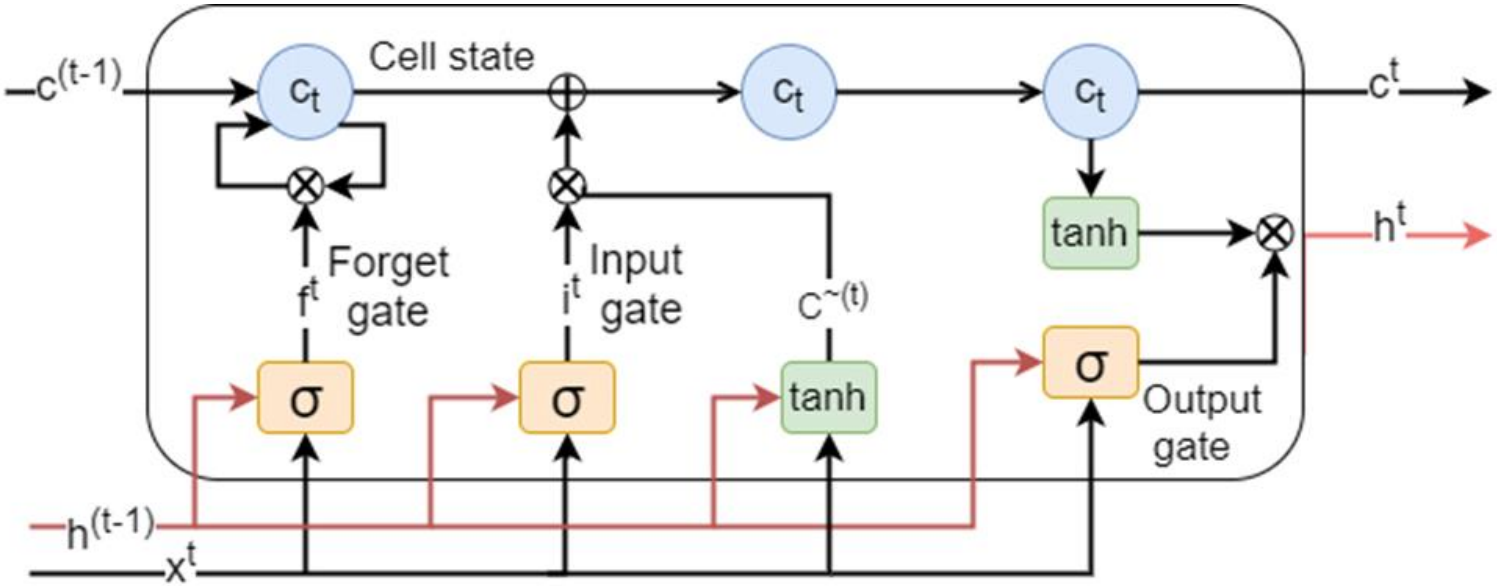
Recurrent Neural Network



- Like CNN, RNN was constructed to solve certain problems that the traditional neural network falls short of.
- The output of a node will be used as input for previous layers or nodes of the same layer.

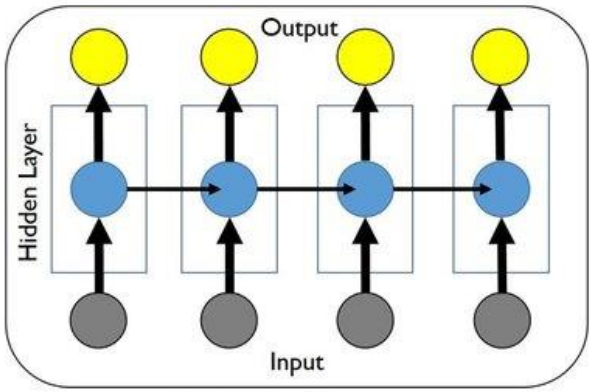
Recurrent Neural Network

- Long Short-Term Memory (LSTM) architecture



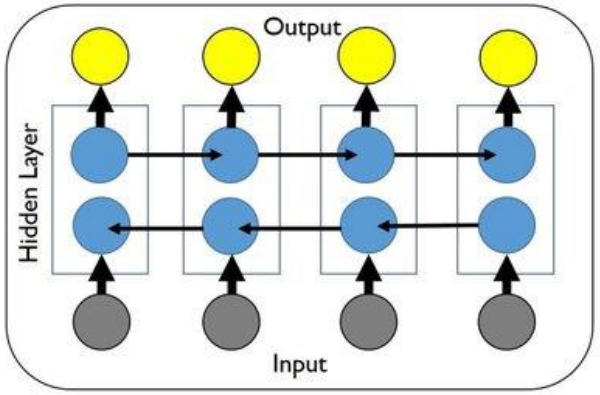
Recurrent Neural Network

- An extension of LSTM is the bidirectional LSTM, or biLSTM



LSTM Architecture

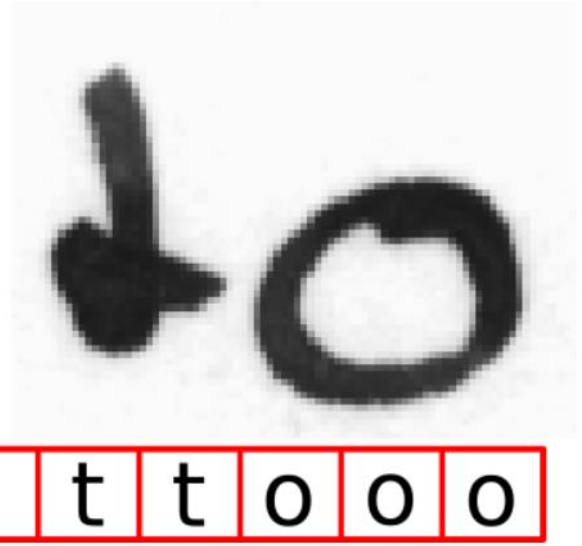
Hochreiter & Schmidhuber, 1997



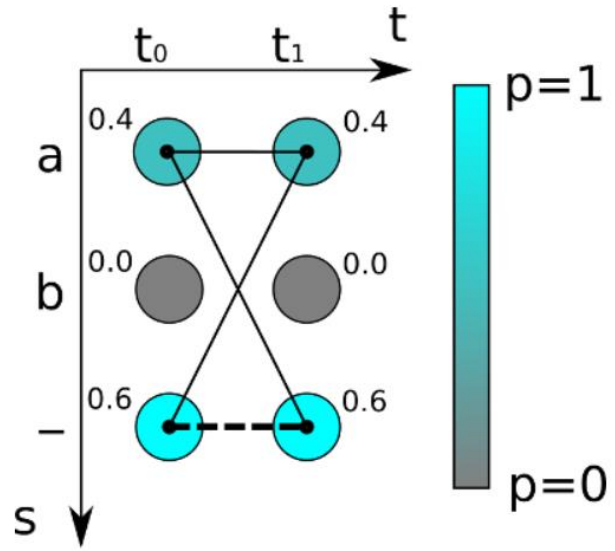
BiLSTM Architecture

Graves & Schmidhuber, 2005

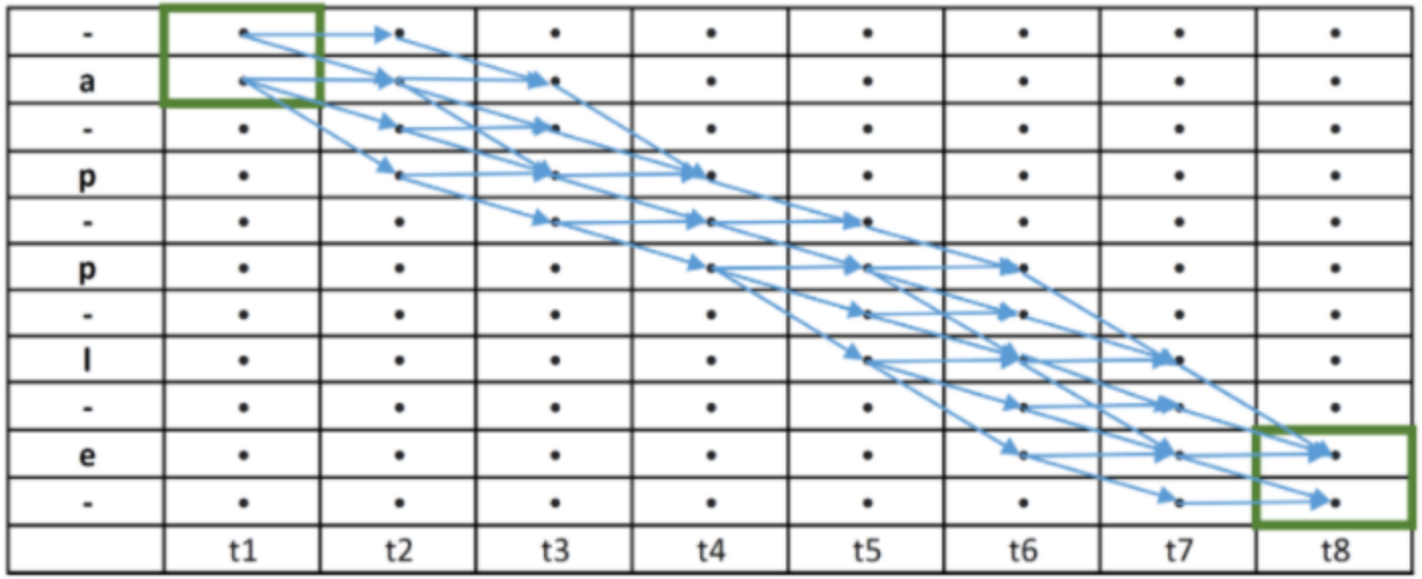
Connectionist Temporal Classification Loss



- In CTC, the task is treated as a multi-class prediction problem, with target cardinality equal to the alphabet plus one blank token.
- Since the output is a matrix of probability between each element and each of the classes, the ground truth score is calculated by summing the probabilities of all combination sequences that return the ground truth.



CTC Example

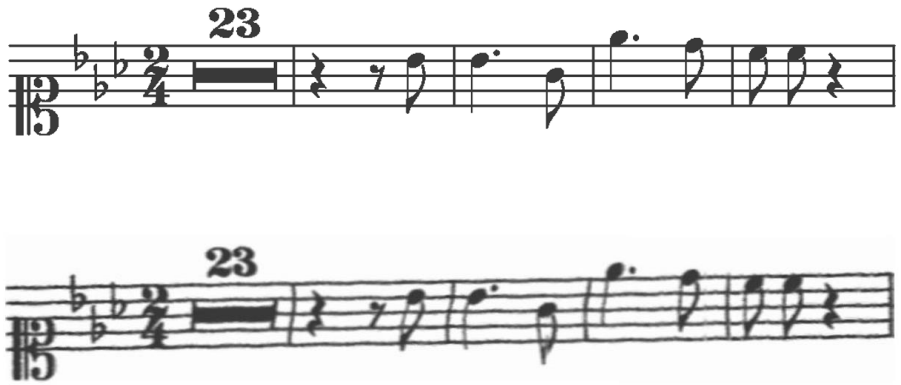


For the alphabet {a,e,l,p}, ground truth “apple”, and sequence of length 8, each dot is a value of the output and the score is calculated by the sum of probability of all blue paths.

DATASET

PrIMuS Dataset

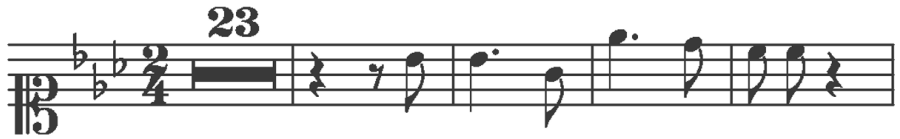
- Consists of only single-staff, monophonic scores
- We chose the updated dataset, which includes a distorted version for each of the images

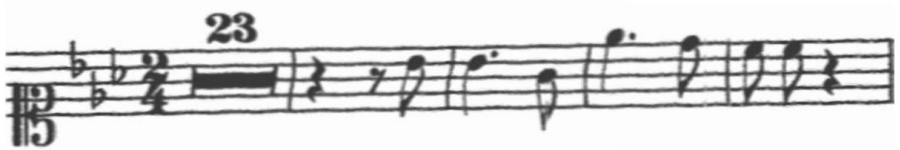


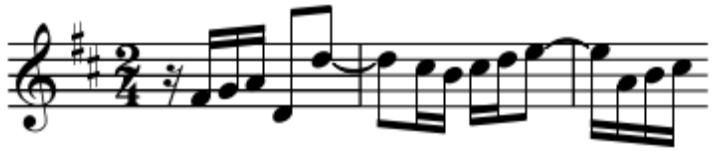
%G-2@2/4$xFCü 6-{'FGA}8{D'D+}/{D6C'B}{''CD8E+}/{6E'AB''C}

a

Images







b

Labels

```
<mdiv>
    <score>
        <score key.sig="2s" meter.count="2" meter.unit="4">
            <staffGrp>
                <staffDef clef.shape="G" clef.line="2" n="1" lines="5" />
            </staffGrp>
        </score>
        <section>
            <measure>
                <staff n="1">
                    <layer n="1">
                        <rest dur="16" />
                        <beam>
                            <note dur="16" oct="4" pname="f" />
                            <note dur="16" oct="4" pname="g" />
                            <note dur="16" oct="4" pname="a" />
                        </beam>
                        <beam>
                            <note dur="8" oct="4" pname="d" />
                            <note dur="8" oct="5" pname="d" tie="i" />
                        </beam>
```

c

```
clef.G-L2, accidental.sharp-L5, accidental.sharp-S3, digit.2-L4, digit.4-L2, rest.sixteenth-L3,
note.beamedRight2-S1, note.beamedBoth2-L2, note.beamedLeft2-S2, note.beamedRight1-S0,
note.beamedLeft1-L4, slur.start-L4, barline-L1, slur.end-L4, note.beamedRight1-L4,
note.beamedBoth2-S3, note.beamedLeft2-L3, note.beamedRight2-S3, note.beamedBoth2-L4,
note.beamedLeft1-S4, slur.start-S4, barline-L1, slur.end-S4, note.beamedRight2-S4,
note.beamedBoth2-S2, note.beamedBoth2-L3, note.beamedLeft2-S3
```

e

```
clef-G2, keySignature-DM, timeSignature-2/4, rest-sixteenth, note-F#4_sixteenth,
note-G4_sixteenth, note-A4_sixteenth, note-D4_eighth, note-D5_eighth, tie, barline,
note-D5_eighth, note-C#5_sixteenth, note-B4_sixteenth, note-C#5_sixteenth,
note-D5_sixteenth, note-E5_eighth, tie, barline, note-E5_sixteenth,
note-A4_sixteenth, note-B4_sixteenth, note-C#5_sixteenth
```

d

IMPLEMENTATION

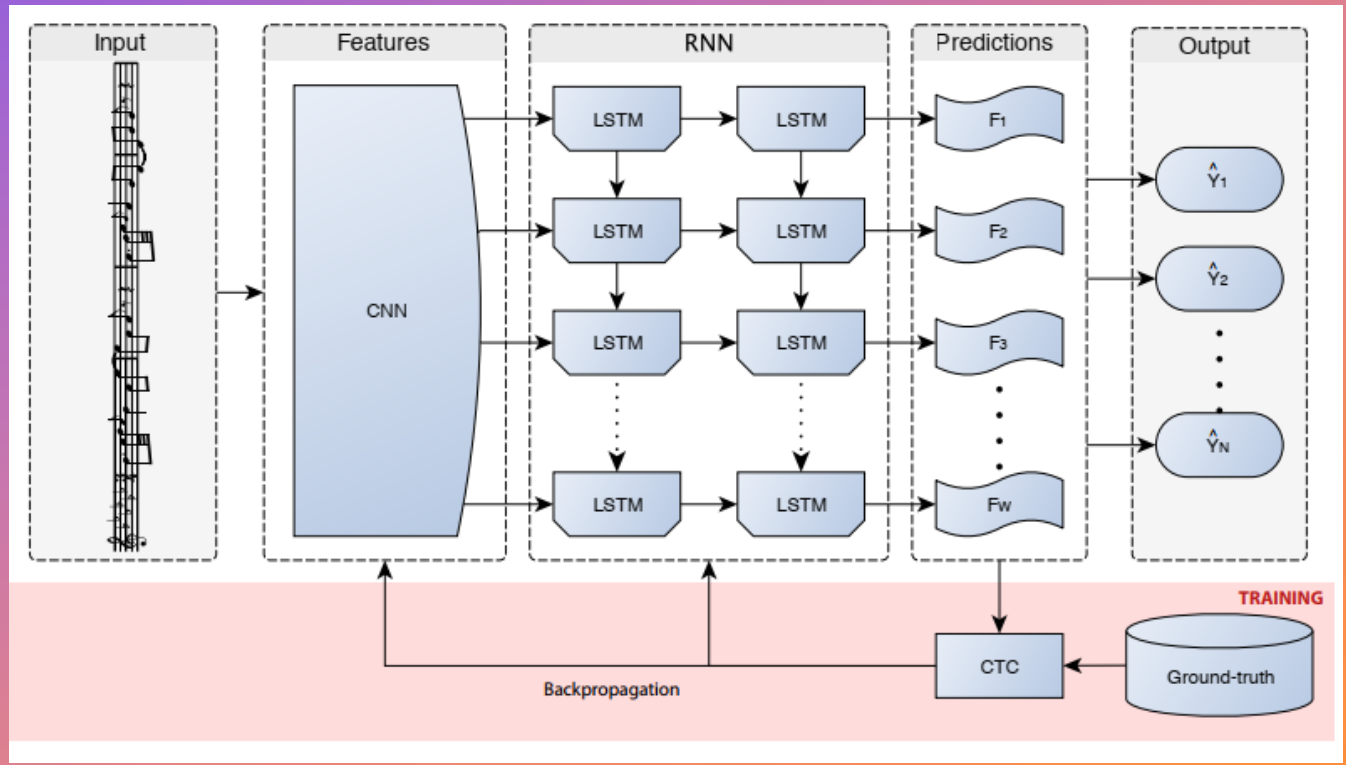
Data preprocessing

- OpenCV 4.0.1 is used to open and scale images

- The images vary in length and need to be padded to the length of the largest one
- The label of each image needs to be converted to a sequence of numerical values

CRNN MODEL

CNN and RNN architectures are combined to make the Convolutional Recurrent Neural Network (CRNN)



CRNN Model

- For optimization, we used the Adam algorithm.
- There are two distinct models: semantic and agnostic recognition

Agnostic model

```
CRNN(
  (cnn): Sequential(
    (conv0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm0): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu0): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv1): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu1): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu2): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu3): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (rnn): Sequential(
    (BiLSTM0): BidirectionalLSTM(
      (rnn): LSTM(256, 256, dropout=0.5, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=256, bias=True)
    )
    (BiLSTM1): BidirectionalLSTM(
      (rnn): LSTM(256, 256, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=759, bias=True)
    )
  )
  (activation): LogSoftmax(dim=2)
)
```

Sematic model

```
CRNN(
  (cnn): Sequential(
    (conv0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm0): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu0): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv1): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu1): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu2): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu3): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (rnn): Sequential(
    (BiLSTM0): BidirectionalLSTM(
      (rnn): LSTM(256, 256, dropout=0.5, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=256, bias=True)
    )
    (BiLSTM1): BidirectionalLSTM(
      (rnn): LSTM(256, 256, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=1782, bias=True)
    )
  )
  (activation): LogSoftmax(dim=2)
)
```

Input ($128 \times W \times 1$)
Convolutional Block
$\text{Conv}(32, 3 \times 3), \text{MaxPooling}(2 \times 2)$
$\text{Conv}(64, 3 \times 3), \text{MaxPooling}(2 \times 2)$
$\text{Conv}(128, 3 \times 3), \text{MaxPooling}(2 \times 2)$
$\text{Conv}(256, 3 \times 3), \text{MaxPooling}(2 \times 2)$
Recurrent block
$\text{BLSTM}(256)$
$\text{BLSTM}(256)$
$\text{Dense}(\lvert\Sigma\rvert + 1)$
$\text{Softmax}()$

Multitask Learning

- We hypothesized that both recognition models will benefit from the same feature extraction layers
- The parameters of this model are not different
- Prone to overfitting

```
CRNN(
  (cnn): Sequential(
    (conv0): Conv2d(1, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm0): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu0): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling0): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv1): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu1): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv2): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu2): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv3): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (batchnorm3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu3): LeakyReLU(negative_slope=0.2, inplace=True)
    (pooling3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (rnn_sem): Sequential(
    (BiLSTM0): BidirectionalLSTM(
      (rnn): LSTM(256, 256, dropout=0.5, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=256, bias=True)
    )
    (BiLSTM1): BidirectionalLSTM(
      (rnn): LSTM(256, 256, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=1782, bias=True)
    )
  )
  (rnn_agn): Sequential(
    (BiLSTM0): BidirectionalLSTM(
      (rnn): LSTM(256, 256, dropout=0.5, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=256, bias=True)
    )
    (BiLSTM1): BidirectionalLSTM(
      (rnn): LSTM(256, 256, bidirectional=True)
      (embedding): Linear(in_features=512, out_features=759, bias=True)
    )
  )
  (activation): LogSoftmax(dim=2)
)
```

RESULT

Evaluation Metrics

- Accuracy(%): Sequences without error
- Symbol error rate(%): Modifications required to produce ground truth from prediction
- Symbol confusion rate(%): Modifications performed on both the symbols and the sequence.

Normal Dataset

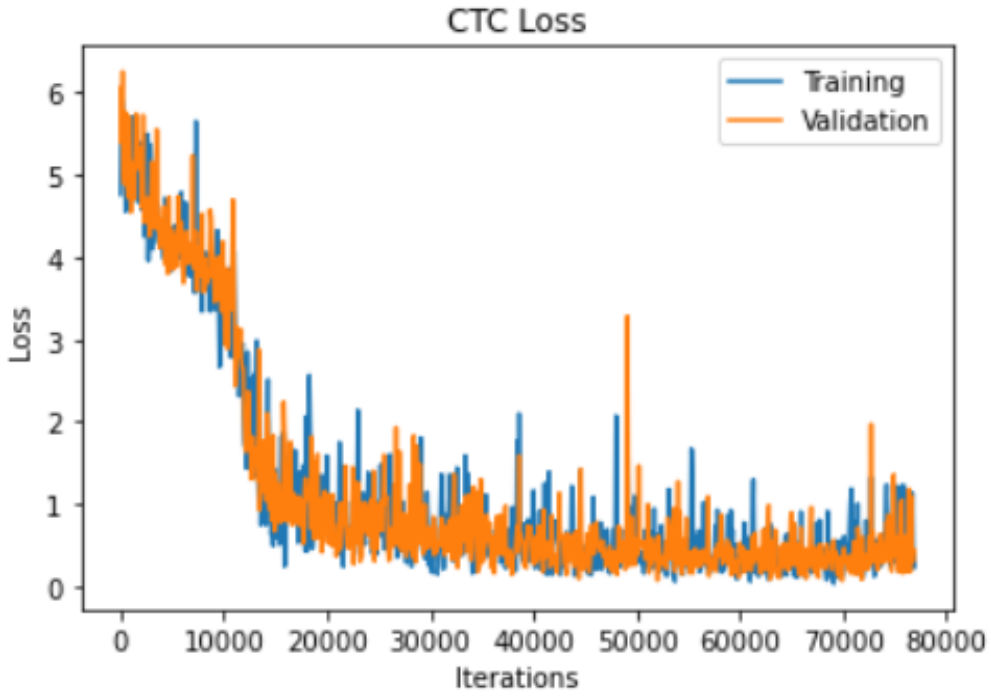
- We could not produce the results as good as the baseline model, due to the difference in batch size (4 vs 16) and epochs (5 vs 16)

	Representation	
	Agnostic	**Semantic**
Sequence Error Rate (%)	17.9	12.5
Symbol Error Rate (%)	1.0	0.8

Normal Dataset

Semantic Result

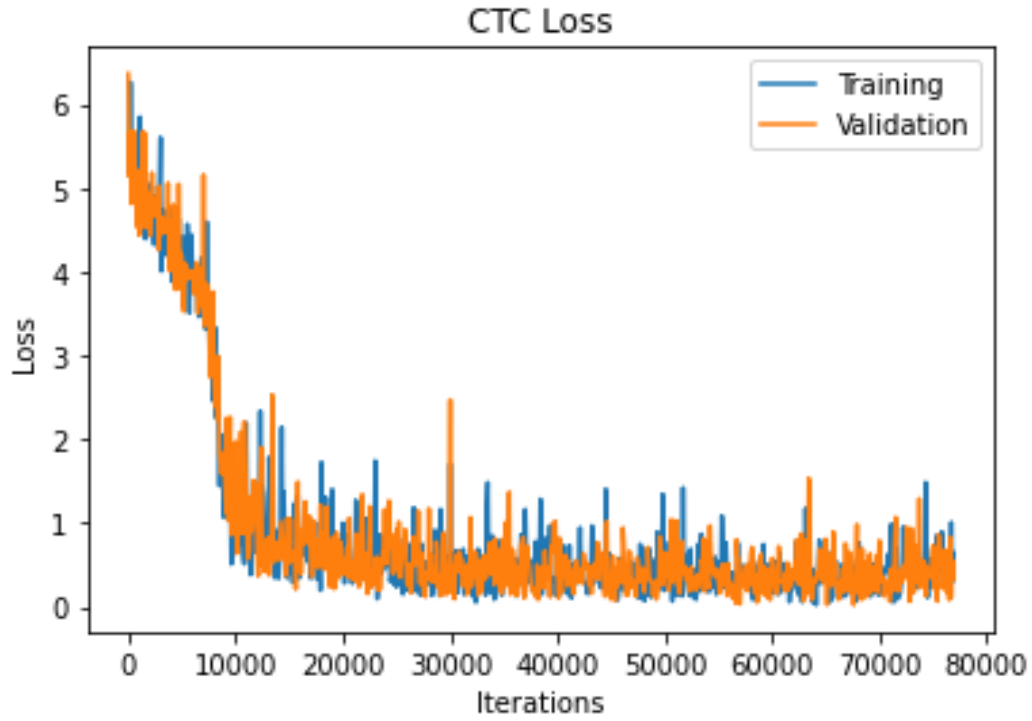
- Accuracy: 24.24%
- Symbol error rate: 10.14%
- Symbol confusion rate: 4.17%



Normal Dataset

Agnostic Result

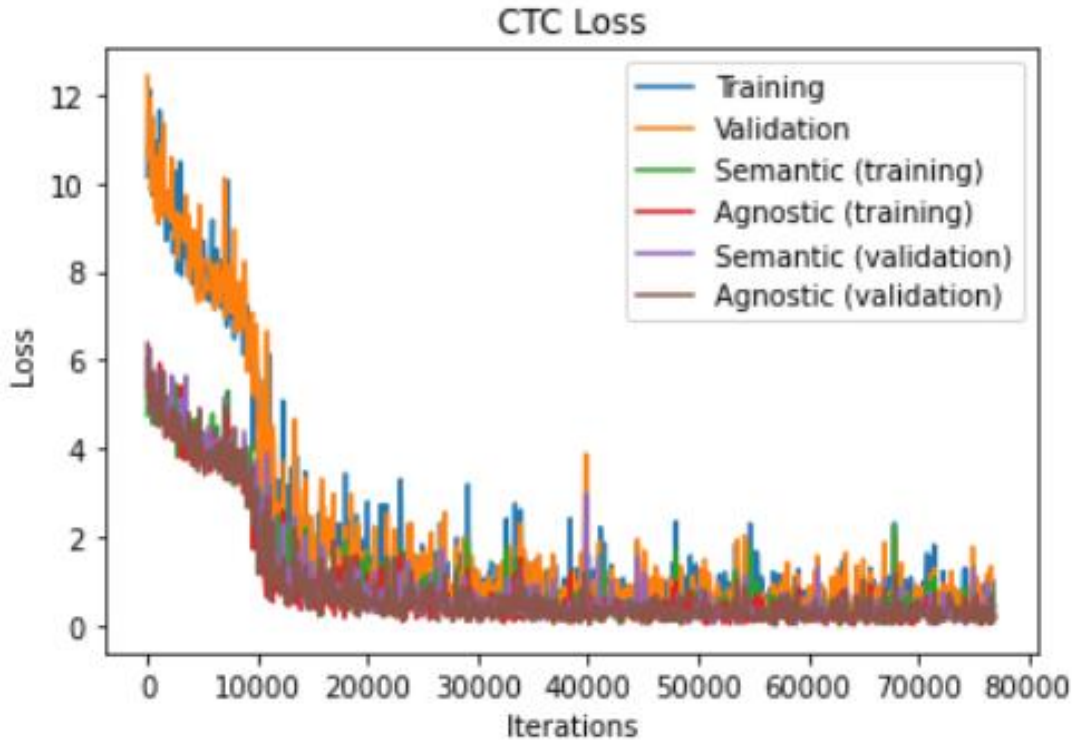
- Accuracy: 18.98%
- Symbol error rate: 14.13%
- Symbol confusion rate: 8.41%



Normal Dataset

Multitask learning

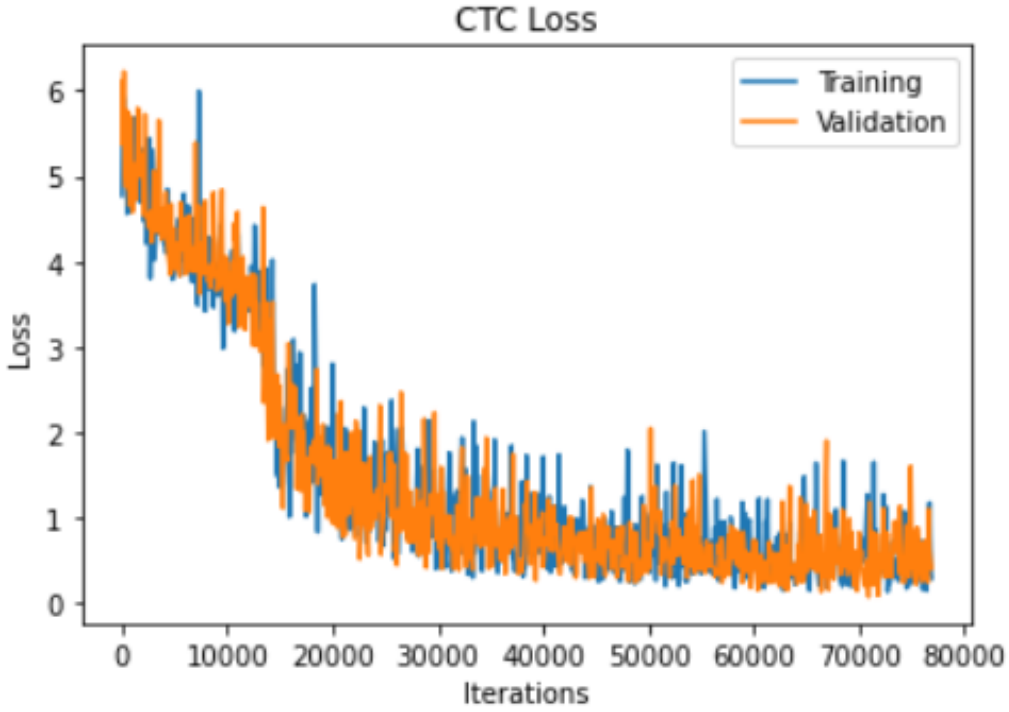
- Accuracy: 20.29%
- Semantic accuracy: 27.67%
- Semantic Symbol error rate: 9.40%
- Semantic Symbol confusion rate: 3.91%
- Agnostic accuracy: 32.15%
- Agnostic symbol error rate: 9.13%
- Agnostic symbol confusion rate: 4.89%



Distorted Dataset

Semantic Result

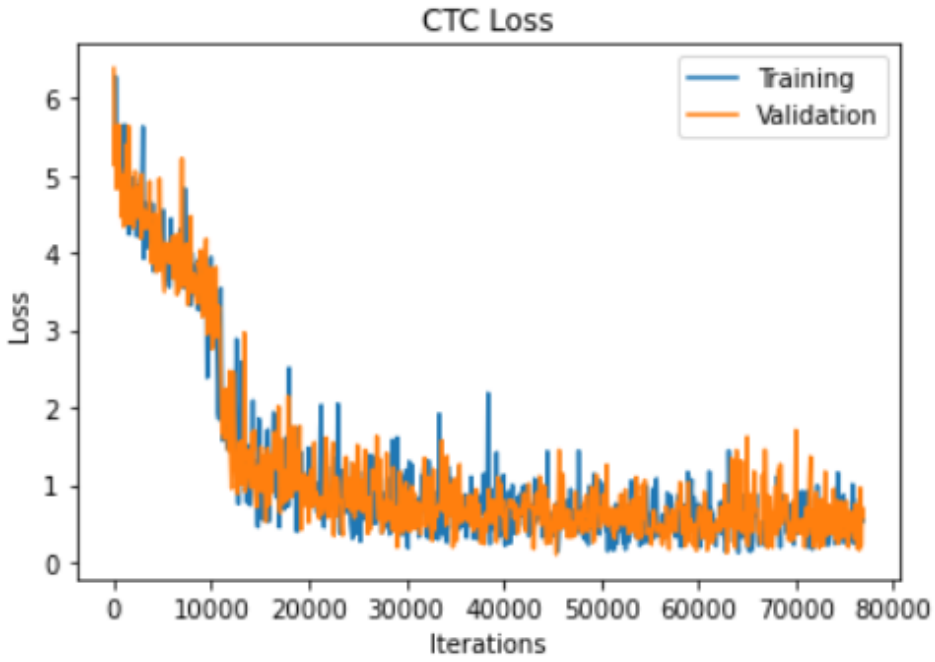
- Accuracy: 17.96%
- Symbol error rate: 14.20%
- Symbol confusion rate: 7.50%



Distorted Dataset

Agnostic Result

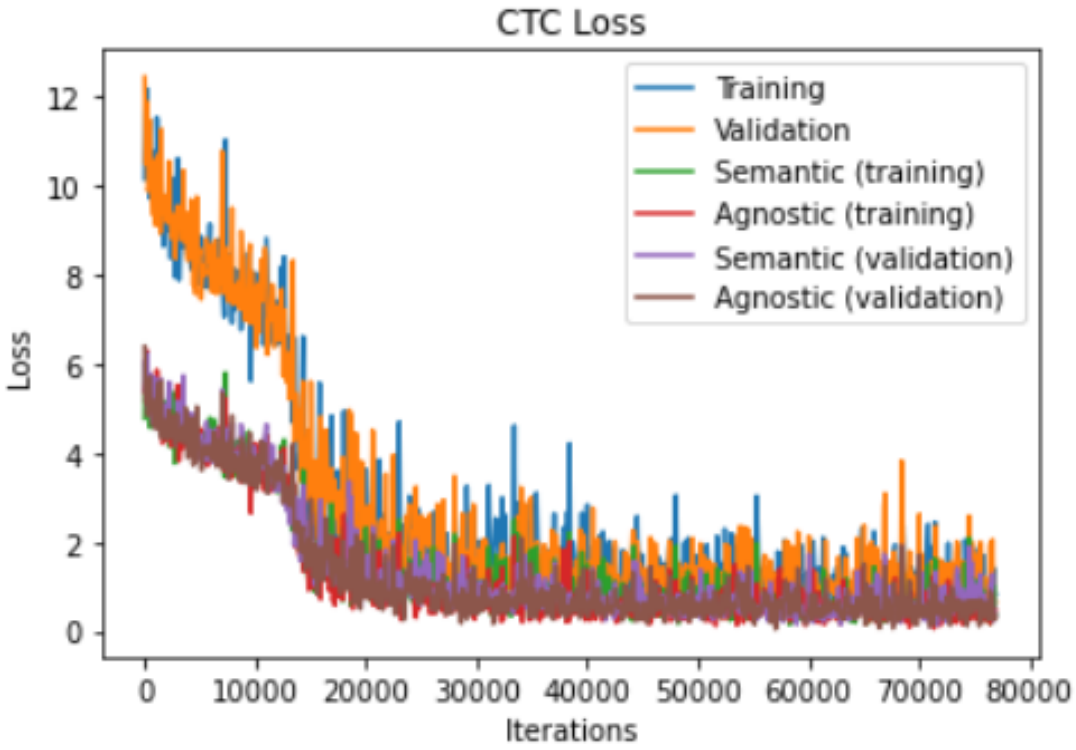
- Accuracy: 8.51%
- Symbol error rate: 18.20%
- Symbol confusion rate: 12.88%



Distorted Dataset

Multitask learning

- Accuracy: 6.17%
- Semantic accuracy: 14.01%
- Semantic Symbol error rate: 15.15%
- Semantic Symbol confusion rate: 6.99%
- Agnostic accuracy: 14.29%
- Agnostic symbol error rate: 14.94%
- Agnostic symbol confusion rate: 9.74%



THANKS FOR LISTENING