Quantization of CNN-based Language Models

Abhishek Yanamandra
College of Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
ayanaman@andrew.cmu.edu

Karthik Pansetty
College of Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
kpansett@andrew.cmu.edu

Robert Haber
Language Technologies Inst.
Carnegie Mellon University
Pittsburgh, PA 15213
rhaber@andrew.cmu.edu

Urvil Kenia
College of Engineering
Carnegie Mellon University
Pittsburgh, PA 15213
ukenia@andrew.cmu.edu

Abstract

In this project, we explore the effect of quantization techniques on CNN-based language models. We primarily use the HW4P2 dataset where the inputs are spectrograms and the outputs are the corresponding English transcripts, and train a Wav2Letter CNN architecture on the dataset before compressing the model with post-training dynamic quantization in ONNX runtime. Our experiments indicate that these techniques can be powerful, but that caution must be exercised when performing the quantization, as naive full-model quantization may lead to extreme performance deterioration. We demonstrate that quantization can be used to achieve a model with a 4x reduction in size with only a 2% loss in performance. These results suggest the potential feasibility of using complex CNN-based language models on resource-constrained devices. Future studies will need to leverage frameworks that harness int-8 op hardware in order to realize the decreased inference times benefit in addition to compression.

1 Introduction

The goal of this project is to enable AI engineers to create fast and compact speech recognition models. This study is motivated by two observations: quantization performance has shown promising result on convolutional neural networks (CNNs), and language models are among those greatest in need of improved performance. However, quantization of recurrent neural network architectures is not well understood. Therefore, we analyze the viability and efficacy of CNN quantization techniques using a CNN-based language model, wav2letter. There are multiple challenges to this research and development. First, we implement and train a float32 version of wav2letter, focusing on the hw4p2 dataset. Second, we quantize the float32 model, thereby resulting in an int8 model. Finally, we compare validation performance between the float32 and int8 models. Hardware limits the scope of the findings in this research, where processors or GPUs that have quantization support are limited. Processors with Intel Deep Learning Boost (also known as AVX-512 VNNI instructions), show promise in enabling quantization performance with int8 operations in their instruction set. Currently, quantization is an active area of development in all the major frameworks, and, given the current software available, we expected to see the most improvements in model size and memory bandwidth. We compare the metrics of accuracy, latency, memory bandwidth, and model size between the float32 and int8 models. Future implications of this work are discussed and directions for improvement are recommended.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

2 Literature Review

Traditional speech recognition models extensively relied on feature extraction, acoustic models and Hidden Markov Models (HMMs). However, the use of deep learning in speech recognition significantly improved and outperformed these traditional models. Models such as DeepSpeech leverage end-to-end deep learning pipelines using recurrent neural networks which result in higher capacity by learning from large datasets to improve performance. Their second version DeepSpeech2 consists of initial convolutional layers and an RNN network with multiple layers.

Wav2letter is an end-to-end CNN-based model for speech recognition developed by Facebook AI Research [Collobert et al., 2016]. It considers MFCCs, power-spectrum, and raw-wav inputs. While the most popular criterion is the CTC criterion, Wav2Letter was trained with a newly proposed ASG criterion. The model is composed of 10 convolution layers, followed by 2 fully-connected linear layers, with the final layer outputting a score for each letter.

Wav2letter++, also developed by Facebook AI Research, is the fastest open-source deep learning speech recognition framework (by a factor of 2). The framework utilizes the ArrayFire tensor library to achieve its impressive performance, and supports several end-to-end sequence models, each composed of a network and some criterion (it currently supports CTC, ASG, and S2S).

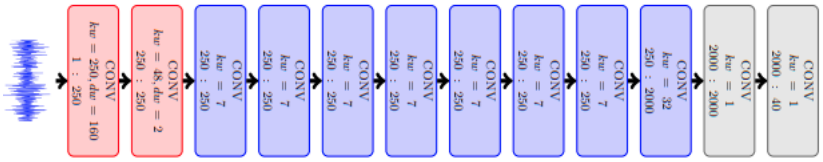

Figure 1: Architecture of wav2letter (First two layers are convolutions with strides and last two layers are fully connected layers). kw is the kernel side and dw is the stride.

Quantization aims to compress the size of deep neural networks by storing model parameters in compact formats, such as integers or even binary numbers instead of 32-bit floating point numbers. The technique was proposed in the 1990s [Fiesler et al. 1990] as a method for simplifying the implementation of neural networks on hardware, and is becoming increasingly popular due to the ever increasing sizes of deep neural network models and the need to deploy models on devices with tight resource constraints.

With quantized networks, forward and backward propagation operations can be conducted with bitwise operations rather than floating point operations. A simple example that illustrates the power of quantization is the computation of the dot product for two binary vectors a and b, which can be computed by taking the bitcount of a and b. When the operations are bitwise, they are carried out by the arithmetic logic unit (ALU), which uses much less energy than a floating-point unit (FPU) [Guo 2018].

In the commonly used fixed-point quantization [Courbariaux et al., 2014], floating point values of the weights and/or activations are approximated by using a set of integers, a scaling factor, and a zero point offset. This method introduces quantization noise, which leads to ranges of degradation in model performance. Many approaches have been proposed to mitigate this noise. One approach is vector quantization [Gong et al., 2014], which involves applying k-means clustering to the network weights and replacing the weight vectors with their corresponding centroids during inference. Though it is simple to implement and has comparable performance to full-precision networks, the accuracy loss caused by k-means clustering cannot be controlled, and no compression ratio constraint is applied to the model [Guo, 2018]. Other approaches have included binarizing [Courbariaux et al., 2015] and ternarizing [Zhu et al., 2016] networks, which constrain the network weights to only two (or three) possible values, and replace many multiplications with bit-shift operations [Choudhary et al., 2020]. These methods have been shown to achieve comparable performance to full-precision networks, but have the engineering challenge of needing to rework the network architecture and retraining from scratch with a special focus on quantization.

As an intermediate between fully reworking network architectures to be quantization-friendly and having a simple API call to quantize a pre-trained model is the method proposed by [Zhou et al., 2017] which showed that weights can be refined by an additional post-quantization training step. With the incremental network quantization method (INQ), the weights in each layer of a pre-trained CNN are divided into two disjoint groups, where the weights in the first group form a low-precision base, and the weights in the second are retrained in an iterative fashion until all weights are converted into low-precision ones (to compensate for the loss from quantization).

More recent studies have focused on data-free quantization, where an already trained network can be quantized for fast inference without any retraining, fine-tuning, or hyperparameter optimization with respect to quantization. One such study [Krishnamoorthi, 2018] proposes a per-channel quantization method where the weights of a convolution tensor are quantized on a per-channel basis. This approach brought the quantized network's accuracy to within 2% of the original network's accuracy for a variety of CNN architectures. The other, [Nagel et. al 2018], attempting to build on Krishnamoorthi by making computation easier for hardware by eliminating the need to compute offset values for each individual channel, introduces a method of data-free quantization through equalizing weights and correcting biases, which achieves nearly the same accuracy as the original network. Even still, data-free quantization does not generalize to all models. Krishnamoorthi found that using quantization-aware training can reduce the accuracy gap to 1% for a variety of models. For quantization-aware training, weights and activations are "fake quantized" during training, in that float values are rounded to int8 values, but the computations themselves are still performed with floating point numbers. This allows weight adjustments to be made during training while aware of the fact that the model will be quantized upon the conclusion of training.

Another proposed quantization method in a latest study [Sean I. Young et al., 2020] is transform quantization, which leverages data-free quantization and takes into account the joint statistics of weights and activations. Transforming a CNN using this method is equivalent to a dimensionality reduction technique such as PCA, while quantizing the insignificant kernels to 0 is equivalent to pruning the weights.

3 Dataset Description

3.1 Source

The data source used for this project is the hw4p2 dataset, which is a collection of English audio recordings with transcription labels. This data is hosted on Kaggle by the 11-785 Spring 2021 course team. We chose this data source due to its size and our familiarity with the set, allowing us to streamline development. The audio recordings are spectrograms with 40 frequency bands at each frame, and the corresponding text labels are given as a list of strings.

Link to HW4P2 Write-up:

`https://deeplearning.cs.cmu.edu/S21/document/homework/HW4/P2/2021spring_HW4P2.pdf`

Link to HW4P2 Dataset:

`https://www.kaggle.com/c/11785-spring2021-hw4p2/data`

Link to HW3P2 Write-up:

`https://deeplearning.cs.cmu.edu/S21/document/homework/HW3/hw3p2_writeup.pdf`

Link to HW3P2 Dataset:

`https://www.kaggle.com/c/11785-spring2021-hw3p2/data`

3.2 Dataset Preprocessing

The spectrogram inputs are already in a processed, feature extracted form from raw wav audio files. Specifically, audio preprocessing steps like: Windowing, STFT, Power Spectrogram, and Mel-scaling

have been applied to the raw audio files. No additional preprocessing of the inputs are required. The text labels require tokenization, creating a list of characters that is then converted to a list of integers. In the HW4P2 data, there are 28,539 training samples and 2,703 test samples where each processed sample is a spectrogram and a normalized English transcript corresponding to that spectrogram.

Some Samples from train and test test along with their transcripts:

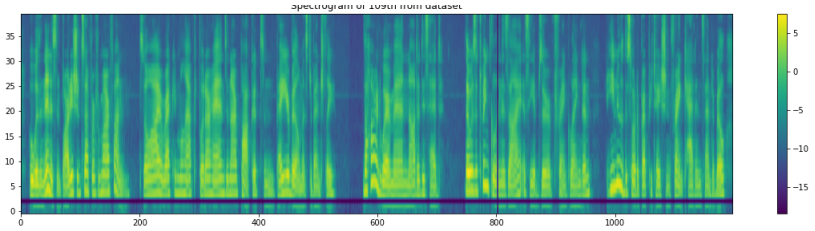

Figure 2: Sample Spectrogram: Text = "who was an innocent party should suffer from our fun so i reckon we'll have to put our hands in our pockets and pay your bill mister benchley the hardware man nodded his head there was a twinkle in his eye as he observed frank langdon he knew the sort of reputation frank had in centerville"

Wav2Letter uses character-level labels where each sentence's base characters are used as labels for the predictions. Base tokens present in the transcripts are listed below.

Base Tokens in our transcripts: Character and ID below

'<s>'	'a'	'b'	'c'	'd'	'e'	'f'	'g'	'h'	'i'	'j'	'k'	'l'	'm'
0	1	2	3	4	5	6	7	8	9	10	11	12	13
'n'	'o'	'p'	'q'	'r'	's'	't'	'u'	'v'	'w'	'x'	'y'	'z'	'-'
14	15	16	17	18	19	20	21	22	23	24	25	26	27
'''	'.'	'_'	'+'	' '	'</s>'								
28	29	30	31	32	33								

So an example sentence "Hello World" is tokenized as ['h', 'e', 'l', 'l', 'o', ' ', 'w', 'o', 'r', 'l', 'd'] and is converted to the respective token ids ([8, 5, 12, 12, 15, 32, 23, 15, 18, 12, 4]) while feeding to the model because ML models can't understand characters as is.

3.3 Data Statistics and Exploration

Utterance length is an important factor we need to take account for, because if the utterance is too large the model would suffer to pickup the right path for the sentence, if the utterance is too short, it will suffer by not pickup richness in the acoustic features.

Subset	Duration	No. of samples
Train	80.5 hrs	28,539
Test	4.3 hrs	2703

The utterance length histograms are as follows for the train data and test data.

4 Design Methodology

This design of this study has four core methodological components: preparing quantized model, forward propagation, loss calculation, and evaluation. In addition to these, other core aspects of a deep learning workflow are integrated, such as optimization and cross validation. An overview of the four core components are provided in diagram form.

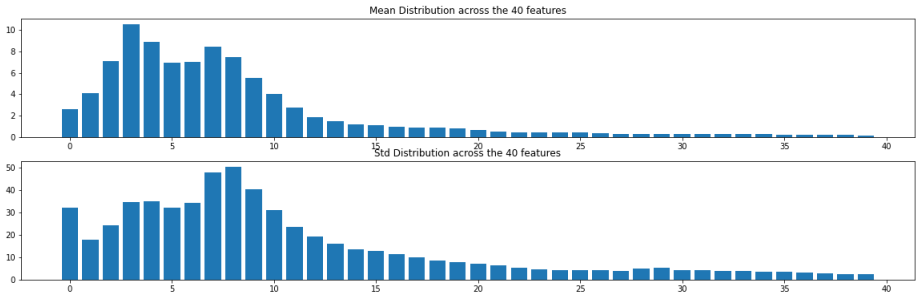

Figure 3: Distribution of Mean and Std across the 40 mel features

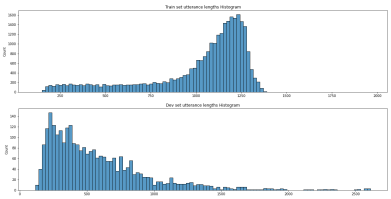

Figure 4: Utterance lengths for Train(avg=1015.705) and Test(avg=574.659) datasets

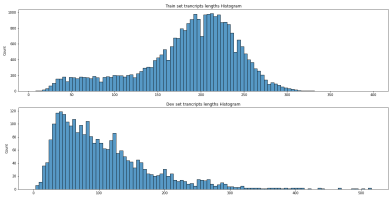

Figure 5: Transcript lengths for Train (avg=184.6) and Test (avg=106.7) datasets

4.1 Model

The model at the core of this work is Wav2Letter - a CNN architecture. This is a many-to-many mapping, which means for each input sequence we are generating a sequence of outputs. For each input, there is one 3D tensor(batch_size, input_length, 40) in the form of spectrogram audio samples and a 2D tensor (batch_size, output_lengths) for the ground truth labels.

We used 4 workers for the data loader with default sampler and implemented our custom collation function in the data loader to pad the sequences with different input and output lengths. The pseudocode for the collation function [1] is:

Algorithm 1 Pseudocode for custom collation function

Require: x : Spectrograms
Require: y : Transcripts
  x_lens ← [len(seq) for seq in x]
  x ← pad_sequence(x)
  y_lengths ← [len(seq) for seq in y]
  y ← pad_sequence(y)
  return x, y, x_lengths, y_lengths

The tabular summary of the model along with is also provided for greater detail.

4.2 Loss Calculation

There is one criterion, CTC Loss, that is being optimized for in this CNN. As a general rule, low loss for this criterion is preferred. The CTC loss is based on maximum likelihood. Minimizing this loss maximizes the log-likelihoods of the target labels. The CTC loss considers all possible sequences

[1] `https://github.com/ukenia/quantization-cnn`

Table 1: Model Summary

Layer (type)	Output Shape	Param #
Conv1d-1	[-1, 250, 50]	480,250
ReLU-2	[-1, 250, 50]	0
Dropout-3	[-1, 250, 50]	0
ConvBlock-4	[-1, 250, 50]	0
Conv1d-5	[-1, 250, 50]	437,750
ReLU-6	[-1, 250, 50]	0
Dropout-7	[-1, 250, 50]	0
ConvBlock-8	[-1, 250, 50]	0
Conv1d-9	[-1, 250, 50]	437,750
ReLU-10	[-1, 250, 50]	0
Dropout-11	[-1, 250, 50]	0
ConvBlock-12	[-1, 250, 50]	0
Conv1d-13	[-1, 250, 50]	437,750
ReLU-14	[-1, 250, 50]	0
Dropout-15	[-1, 250, 50]	0
ConvBlock-16	[-1, 250, 50]	0
Conv1d-17	[-1, 250, 50]	437,750
ReLU-18	[-1, 250, 50]	0
Dropout-19	[-1, 250, 50]	0
ConvBlock-20	[-1, 250, 50]	0
Conv1d-21	[-1, 250, 50]	437,750
ReLU-22	[-1, 250, 50]	0
Dropout-23	[-1, 250, 50]	0
ConvBlock-24	[-1, 250, 50]	0
Conv1d-25	[-1, 250, 50]	437,750
ReLU-26	[-1, 250, 50]	0
Dropout-27	[-1, 250, 50]	0
ConvBlock-28	[-1, 250, 50]	0
Conv1d-29	[-1, 250, 50]	437,750
ReLU-30	[-1, 250, 50]	0
Dropout-31	[-1, 250, 50]	0
ConvBlock-32	[-1, 250, 50]	0
Conv1d-33	[-1, 2000, 51]	16,002,000
ReLU-34	[-1, 2000, 51]	0
Dropout-35	[-1, 2000, 51]	0
ConvBlock-36	[-1, 2000, 51]	0
Conv1d-37	[-1, 2000, 51]	4,002,000
ReLU-38	[-1, 2000, 51]	0
Dropout-39	[-1, 2000, 51]	0
ConvBlock-40	[-1, 2000, 51]	0
Conv1d-41	[-1, 35, 51]	70,035
LogSoftmax-42	[-1, 35, 51]	0
Total params: 23,618,535 Trainable params: 23,618,535 Non-trainable params: 0		
Input size (MB): 0.02 Forward/backward pass size (MB): 9.30 Params size (MB): 90.10 Estimated Total Size (MB): 99.42		

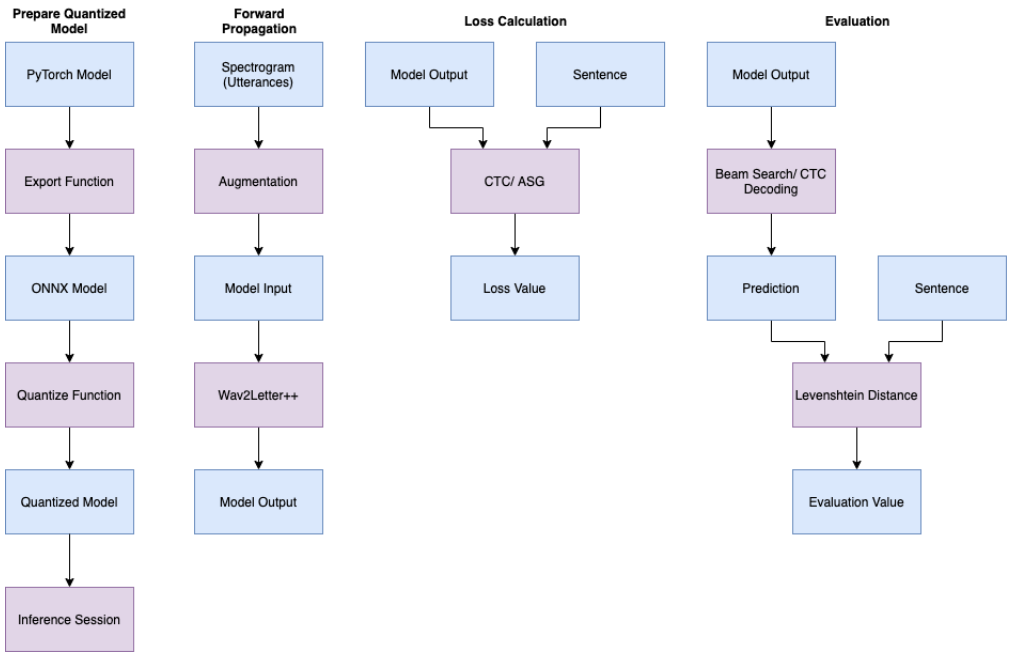

Figure 6: Diagrams for (from left to right) quantization process, forward propagation, computation of loss, and evaluation

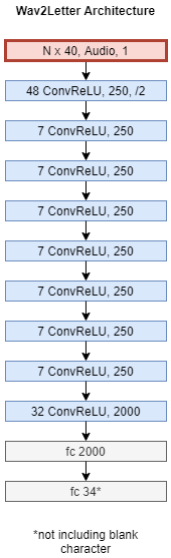

Figure 7: Architecture of wav2letter

and allows for the usage of blank symbols which help in separating repetitions of symbols and are also useful to label the so called "garbage" frames. It uses the log probabilities of the outputs (the characters in the vocabulary including the blank symbol).

The CTC loss is given by the expected divergence:

$$\text{DIV} = -\sum_t \sum_r \gamma(t,r) \log y_t^{S(r)}$$

where $\gamma(t, r)$ is the posterior probability of the character at index r at time t. It is given by

$$\gamma(t,r) = \frac{\alpha(t,r)\beta(t,r)}{\sum_{r'} \alpha(t,r')\beta(t,r')}$$

Here $\alpha(t, r)$ is the forward probability and $\beta(t, r)$ is the backward probability of character at index r at time t.

$$\alpha(t,r) = \sum_{q:S_q \in pred(S_r)} \alpha(t-1,q) y_t^{S_r}$$

$$\beta(t,r) = \sum_{q:S_q \in succ(S_r)} \beta(t+1,q) y_{t+1}^{S_q}$$

4.3 Optimization

For optimizing the model, the Adam optimizer was used. Adam is a first-order gradient based stochastic optimization algorithm, based on adaptive learning rates from estimates of the first and second moments of the parameters' gradients [Kingma et al., 2015]

Algorithm 2 Pseudocode for Adam optimizer [Kingma et al., 2015]

Require: α : Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0 : Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while $\theta_t$not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

4.4 Evaluation

We are using the Levenshtein edit distance to evaluate our model. The Levenshtein edit distance between string a and b is the minimum number of single-character operations (insertions, deletions, or substitutions) to convert a to b.[2] It is formally defined by the $lev_{a,b}(|a|, |b|)$, where

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if } \min(i,j) = 0, \\ \min \begin{cases} \mathrm{lev}_{a,b}(i-1,j) + 1 \\ \mathrm{lev}_{a,b}(i,j-1) + 1 \\ \mathrm{lev}_{a,b}(i-1,j-1) + 1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

[2] `https://www.baeldung.com/cs/levenshtein-distance-computation`

Algorithm 3 Pseudocode for Evaluation

```
Require: test_data: Test dataset utterances
  Criterion ← CTCLoss()
  CTC_decoder ← CTCBeamDecoder(label_map, beam_width=20)

  for (spectrograms, labels) in test_data do
    output_logits ← Model(spectrograms)
    loss += Criterion(output_logits, labels, output_lengths, label_lengths)
    beam_results ← CTC_decoder.decode(output_logits, output_lens)
    transcripts ← []
    for (beam_result) in beam_results do
      transcripts.append(index2letter(max(beam_result)))
    end for
  end for
  for (transcript, label) in (transcripts, labels)  do
    dist += Levenshtein_dist(transcript, (index2letter(label)))
  end for
  avg_loss = loss / len(test_data)
  avg_dist = dist / len(test_data)
  return  avg_loss, avg_dist
```

5 Quantization

ONNX has broadly 3 types of quantization techniques namely dynamic quantization, static quantization and quantization-aware training. [3].

The scale factor and zero point for quantization are calculated as follows:

$$\text{scale\_factor} = \frac{(\text{data\_range\_max} - \text{data\_range\_min})}{(\text{quantization\_range\_max} - \text{quantization\_range\_min})}$$
$$\text{VAL\_fp32} = \text{scale} * (\text{VAL\_quantized} - \text{zero\_point})$$

The zero_point represents the zero in the quantization space.

5.1 Dynamic Quantization

Dynamic quantization is a type of post-training data-free quantization and it computes the scale factor and zero point for the activations dynamically at the time of inference. It computes the range of data at the time of inference to compute the statistics and quantize the activations.

5.2 Static Quantization

Static quantization is another type of post-training quantization which quantizes the weights and activations before inference. It computes the scale factor and zero point using a 'representative' dataset which is usually a subset of the training or validation dataset.

5.3 Quantization Aware Training

Quantization-aware training modifies the training graph to simulate quantization during the forward pass itself thus making quantization error a part of training loss which the optimizer tries to minimize. It can be viewed as the model learning with the knowledge of future quantization.

[3] https://www.onnxruntime.ai/docs/how-to/quantization.html#:~:text=Quantizing%20an%20ONNX%20model&text=Dynamic%20quantization%3A%20This%20method%20calculates,the%20quantization%20parameter%20of%20activations

6 Experiment

We trained our Wav2Letter model on both HW3P2 and HW4P2 data using Data Augmentation of Time Masking and Frequency Masking to prevent overfitting using CTC Loss and performed decoding during inference using a Beam Decoder of Beam width 20. We tried training the model using both Adam and SGD optimizers, Adam turned out to give better results. We then exported the trained model into ONNX runtime and then quantized the model to an int8 model using Post Training Quantization and then performed inference using the int8 model we obtained from the previous step. The layers currently supported on ONNX are Conv, MatMul, MaxPool, Relu, Clip, Add and Mul.

We have focused mainly on dynamic post-training quantization in our experiments. For Post Training Quantization, we quantized the model in multiple variants. We quantized the complete model but that didn't yield very good results (please see figure 9 below). We also quantized a model excluding the first two and last two layers, which gave us very good results (please see figure 9 below) and we also quantized a model excluding just the first layer which gave us the best results in terms of model size with almost no decrease in model performance. We also trained the Wav2Letter model using Quantization Aware Training in PyTorch (figure 9).

We observed good results on HW3P2 data but they could not be replicated for HW4P2 data. This motivated us to investigate the quantization of specific layers. We began with quantizing all layers except the first 2 and last 2 CNNs. This resulted in better performance but did not reduce the model size by a significant margin the reason being they have more parameters. We hypothesized and confirmed that the first CNN layer extracted the maximum information from the data prompting us to quantize all layers except the first CNN layer. This resulted in reducing the model size by a considerable margin which was comparable to the fully quantized (int8) model while resulting in a performance that is comparable to float32 model. This indicated the ideal trade-off between model size and performance and we refer to it as semi-int8 model in our further discussion.

The model pipeline including quantization process, forward propagation, computation of loss, and evaluation are as mentioned in Figure 7.

We implemented a baseline model of the Wav2Letter to perform Speech recognition on utterances using the HW3P2 data and achieved a validation score of 8.92. The hyper-parameters used were: Batch size of 128, Adam optimizer with a weight decay of 1e-5, an initial learning rate of 2e-3 with a ReduceLROnPlateau scheduler with patience of 5 and factor 0.1. We added a dropout layer of p=0.2 after every Conv1d-Relu layer.

On HW4P2 data, we achieved a validation score of 28.96. The hyper-parameters used in this case were: Batch size of 128, Adam optimizer with a weight decay of 5e-5, an initial learning rate of 2e-4 with a ReduceLROnPlateau scheduler with patience of 5 and factor 0.1. We added a dropout layer of p=0.2 after every Conv1d layer.

The summary of the model is included in table 1.

Below in our results, **int8** is the fully quantized model, **semi-int8** is the model where all the layers are quantized but the first layer, **float32** is the base unquantized model and **qat** is quantization aware training quantized model.

6.1 Hardware Specification

The model was trained using AWS EC2 g4dn.2xlarge and inference was performed on AWS EC2 c5.4xlarge for faster inference. Their specifications are as below:

Inference is faster with AVX-512 VNNI instruction set which maximizes use of compute resources, improves cache utilization and avoids potential bandwidth bottlenecks[4] which is an advantage while utilizing Intel CPUs as compared to AMD CPUs.

[4] `https://www.intel.com/content/dam/www/public/us/en/documents/product-overviews/dl-boost-product-overview.pdf`

Table 2: Hardware specification

Purpose	Training	Inference
Instance Type	g4dn.2xlarge	c5.4xlarge
CPU name	Intel (R) Xeon (R) Platinum 8259CL	Intel (R) Xeon (R) Platinum 8275CL
Clock speed	2.5 GHz	3 GHz
vCPU	8	16
CPU Memory	32 GB	32 GB
L2 Cache size	1 MB	1 MB
GPU name	Nvidia (R) T4	-
GPU Memory	16 GB	-
GPU Clock speed	33 MHz	-

6.2 Loss Results

The validation loss increases with the extent of quantization which is as per expectations. The loss value for int8 model is much higher than both the semi-int8 and float32 models.

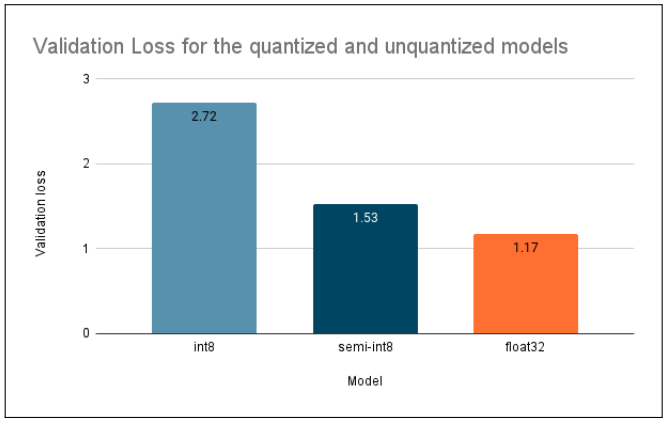

Figure 8: The Loss values for the quantized and unquantized models

6.3 Evaluation Results

The performance on both datasets in terms of model size was as expected and the model size was inversely proportional to the degree of quantization. We did, however, observe sub-optimal results post quantization in terms of the average Levenshtein distance on HW4P2 for int8 and qat (quantization-aware training) models. The semi-int8 model resulted in the best performance on both datasets in terms of model size and average Levenshtein distance.

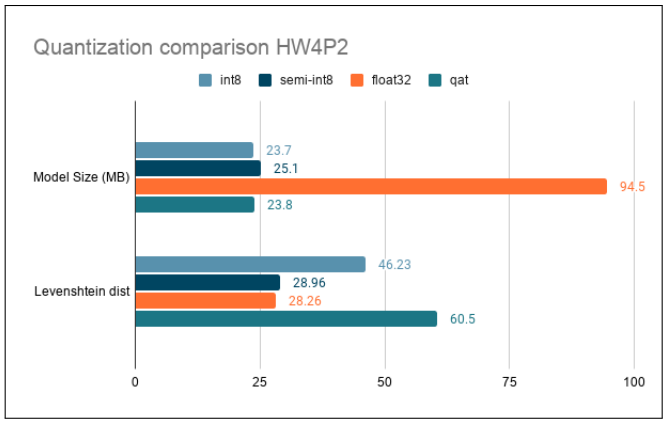

Figure 9: Quantization comparison for HW4 data

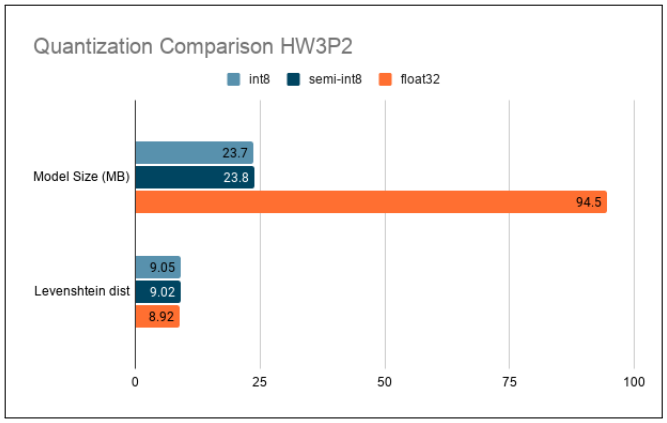

Figure 10: Quantization comparison for HW3 data

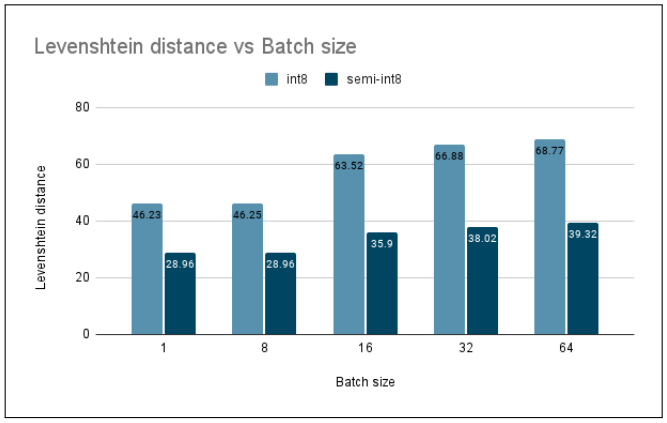

Figure 11: Levenshtein distances using different batch sizes during inference on quantized models

Here, we observe that utterances with higher lengths are doing worse than compared to the ones with smaller lengths.

6.4 Sample Predictions

Semi-Quantized model predictions:

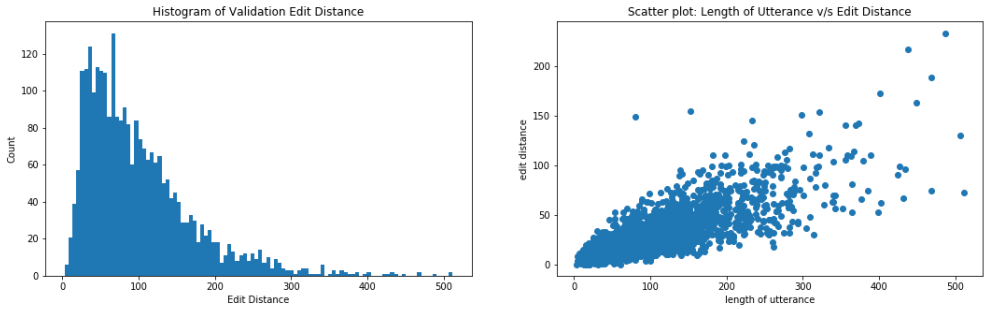

Figure 12: Validation Edit Distance Histogram and Length of Utterance vs Edit distance Scatter Plot

Top-5:

A-1: but for the sight that awaited him he was not prepared at all
P-1: but for the sight that awaited him he was not prepared at all
A-2: she had no companion near her
P-2: she had no compaion near her
A-3: i did not ask i was beyond it
P-3: i did not ask i was beyondit
A-4: this was carried out to the letter
P-4: this was carried out to the lette
A-5: i don't know what becomes of the ladies
P-5: i don't know what be comes of the ladies

Average-5:

A-1: i believe the seriousness of the americans arises partly from their pride
P-1: i believ for sias ness of the imacicans arisest posches from their cried
A-2: the merry andrews and mountebanks of tarrinzeau field were aghast at gwynplaine
P-2: the mary andrers and moun to bente of tudnins of field were a gassit wynplaing
A-3: a snake of his size in fighting trim would be more than any boy could handle
P-3: this make ot i sizles and figding trim would be more than ne boy a handle
A-4: three o'clock came four five six and no letter
P-4: we have not caime bola fi bat and ho letter
A-5: this man was clad in a brown camel hair robe and sandals and a green turban was on his head
P-5: this man was cla in a brown a commo hil lowe and sandlels and a gleemn terven was on hi said

Worst-5:

A-1: he acquiesced in the old distinction of the greek philosophers between the rational and sensitive soul of man that he might reserve the logos for intellectual functions and employ the subordinate human principle in the meaner actions of animal life
P-1: the apliyes in the old istittion of the ree filossiopers between the rastnol attenptid it sol d maner that he mightreservedul wall thust frang geolecual wonction slo i a supvornak hum an principal a ad neiter actions of out of a lifee
A-2: the past he regretted he was discontented with the present and the future he had reason to dread the oriental bishops successively disengaged their cause from his unpopular name and each day decreased the number of the schismatics who revered nestorius as the confessor of the faith
P-2: the pass he ta wried he was sis contenpt with the present of the future gev ers as droaod for ad tortiships successively dis agage their prause were his untalpura man and eac staye a prist the number of thiskis notaxse riveared mysserious is thefusserof the faith
A-3: a very simple little marriage feast but more love good will and tender wishes adorned the plain table than is often found at wedding breakfasts and better than any speech or song was letty's broken whisper as she folded her arms round david's empty chair when no one saw her heaven bless and keep and bring him back to us

P-3: and very simple little marig ches but mor lirte good well ind tender wistes and going tho praine tableven his oftand found ter wetting backfess and better than any stence her son was lhethes speckend wespher as he ford her arms srount he rind o ko tai when ynow one saw her heavern blussin te them brag him back tro us
A-4: dulcinea is perishing thou art living on regardless i am dying of hope deferred therefore untruss thyself with a good will for mine it is here in this retired spot to give thee at least two thousand lashes
P-4: dat soe mol wis poition thor rit gliving mhe hot the hearbes i am dn it hoo te frod therefor untest byself at e giod wat from mia is teyu in this we daa ad to gi bi at least toth the housang dricieus
A-5: it was plain that his castanet girl his mother and sister took a pleasure in crediting her daily with some fresh and unpleasing instrument could have had neither taste money nor honesty to such a point as this
P-5: it was clan at his tust te est of all bo his motter s a sui to the presant cutdit en her beat we some frers in incces r insta but if oenottetaste money not honsi to such a portosess

6.5 Inference Performance

Here, we have observed that the inference times for both the int8 and semi-int8 models are higher than the original models. One reason for this might be because the framework does not fully harness int-8 op hardware in order to realize the decreased inference times benefit in addition to compression.

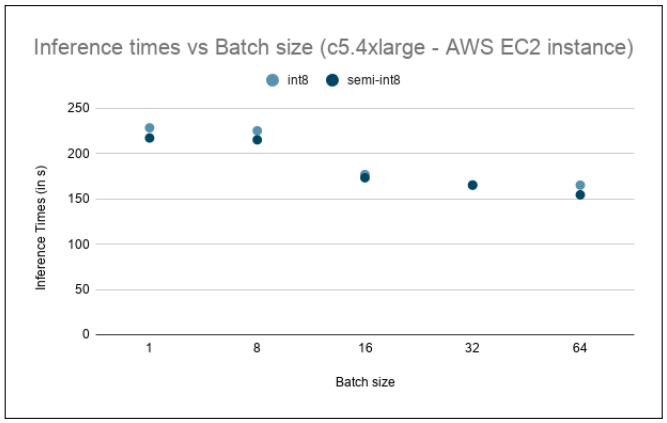

Figure 13: Inference times for the quantized models with different batch sizes

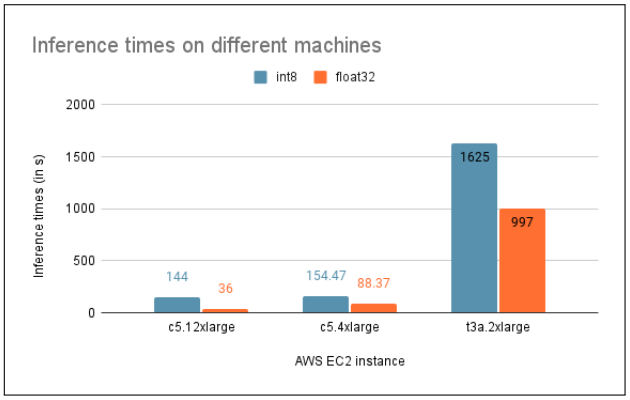

Figure 14: Inference times for the quantized and the unquantized models

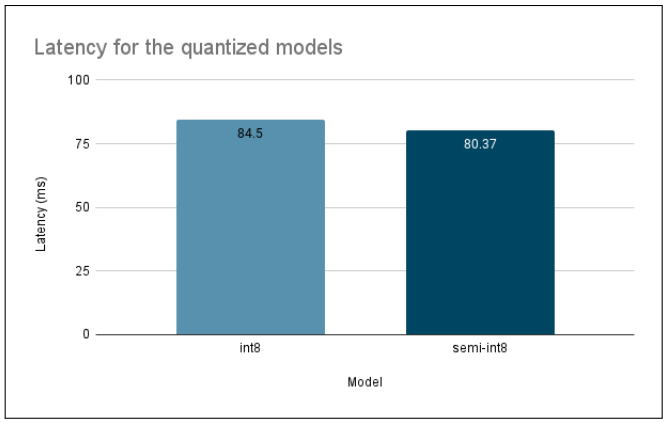

Figure 15: Latency for the quantized models

6.6 Performance comparison on HW4P2 and HW3P2

Naturally, the task required in HW4P2 is more challenging to learn. In HW3P2, the mapping that is learned is from utterance frames to particular phonemes, while in HW4P2, the mapping is from utterance frames to English letters. Two frames that appear nothing alike in a small window will likely correspond to different phonemes, but map to the same letter depending on the spelling of the target word. For example, each 'e' in Mercedes sounds different, so while in HW3P2, each frame will be mapped to different phonemes, in HW4, they all map to the letter 'e'. Analogously, the words 'but' and 'put' contain the letter 'u' but the acoustics are different; the phonetic representations capture the difference (/bət/ and /pʊt/). For the model to learn patterns out of these ambiguous representations, it would likely need to be trained on more samples. Experimenting with larger datasets like Librispeech may narrow the gap between Wav2Letter performance on HW3P2 and HW4P2.

6.7 Observations

As shown in the results, quantization can be useful as it is capable of reducing model size by 4x with nearly the same performance, but naively quantizing all layers can lead to enormous degradation in performance. The first CNN layer contains the most information and quantizing it results in significant drop in performance. Keeping it unquantized, the other layers can be quantized resulting in a 75% size reduction with almost no performance decrease. Implementing a CNN-based architecture resulted in decent results without attention or RNNs. This resulted in much lower in size (by 6x-7x) compared to RNN models and Transformers for the ASR (Automatic Speech Recognition) model. We observed that the inference changes with batch size on the quantized models. We got better results using lower batch sizes. We also observed that c5.4xlarge was much faster for CPU inference than t3a.2xlarge (by 10x-11x) because of AVX-512 VNNI instruction set (combines multiple instructions designed for Deep Learning models). Finding the right layers to quantize in Deep Convolutional models in Acoustic/Language tasks is key for successfully implementing complex CNNs on edge devices.

7 Future Works and Discussion

7.1 Major Challenges

Since we are using the HW4P2 dataset, we do not have the benefit of using the Wav2Letter baseline directly. Instead, we resort to the provided grade cutoffs for the assignment for benchmarking our model performance. Additionally, it is difficult to achieve high performance on this assignment (Levenshtein distance below 20), as we are restricted to using CNNs only, which do not have a mechanism such as attention to capture long term dependencies.

7.2 Project Constraints

Due to the short window of the project, we were not able to run as many experiments as we would have wished. With more time, we would have been able to implement the ASG criterion and its decoder to more closely mirror the paper's implementation, as well as debug quantization-aware training in our system, and implement newer quantization methods (for CNNs specifically) such as transform quantization. Quantization is still in the nascent stages and not all frameworks and not all types of Neural Network layers are supported yet. Another constraint is that, currently there isn't enough documentation for Quantization.

7.3 Areas Needing Improvement

7.3.1 Data

To get a more informed comparison with the Wav2Letter model or quantization as a whole, we would ideally use the LibriSpeech data which is 12.5 times larger (1000hrs) than our present dataset (80hrs). This would help in providing richer acoustic context to our model.

7.3.2 Training

We believe the training performance could be improved by the implementation of ASG criterion in place of CTC. We also believe further experiments with quantization-aware training could help improve results.

7.3.3 Other Shortcomings

Ideally, when using quantization techniques, it is desirable to obtain a significant decrease in model size and inference time simultaneously. Due to the use of dynamic quantization, we were not able to attain a lower inference latency with our quantized model because the chosen framework is not leveraging the hardware architecture. In order to achieve this, more experiments will need to be conducted using static quantization, and observing whether said method could achieve a lower latency without degradation.

7.4 Future Work

We plan to further experiment with a different quantization scheme with 16-bit weights and 8-bit activations as well as implement pruning, quantization and Huffman coding as implemented in the Deep Compression paper. We also plan to leverage static quantization to further reduce inference time and implement transform quantization for improving the accuracy of the quantized model.

8 Division of Work

See Table 3 for the breakdown of how we have been dividing the work.

Table 3: Division of Work

Contributor	Emphasis
Karthik	Implemented Base code, Wav2Letter model, Baselines on HW3 and HW4 data and Quantization Aware Training on PyTorch
Urvil	Dynamic quantization on ONNX, experimented quantization of PyTorch QAT models on ONNX, AWS environment setup, researched quantization schemes and literature
Abhishek	Implemented base evaluation scripts, hypertuned on HW3 and HW4 data Performed Dynamic and QAT quantization on PyTorch
Rob	Quantized models in ONNX, conducted experiments measuring statistics for quantized vs. unquantized model performance, researched quantization approaches and literature

Github Link: `https://github.com/ukenia/quantization-cnn`

9 Acknowledgement

The TA mentor for this project is Joseph Konan, and this proposal is based on a template provided by him.

References

Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A comprehensive survey on model compression and acceleration. Artificial Intelligence Review, pages 1–43, 2020. 10

R. Collobert, C. Puhrsch, and G. Synnaeve, "Wav2Letter: an Endto-End ConvNet-based Speech Recognition System," in arXiv,2016.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In Advances in Neural Information Processing Systems, pages 3123–3131, 2015.

Matthieu Courbariaux, J.-P. David, and Yoshua Bengio. Training deep neural networks with low precision multiplications. arXiv preprint arXiv:1412.7024, 2014

Amodei, Dario, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." In International conference on machine learning, pp. 173-182. PMLR, 2016.

Emile Fiesler, Amar Choudry, and H John Caulfield. Weight discretization paradigm for optical neural networks. In Optical interconnections and networks, volume 1281, pp. 164–174. International Society for Optics and Photonics, 1990

Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. arXiv preprint arXiv:1412.6115v1, 2014.

Yunhui Guo. A survey on methods and theories of quantized neural networks. CoRR, abs/1808.04752, 2018

Hannun, Awni, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger et al. "Deep speech: Scaling up end-to-end speech recognition." arXiv preprint arXiv:1412.5567 (2014).

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In ICLR, 2015.

Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. arXiv preprint arXiv:1806.08342, Jun 2018.

Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. arXiv preprint arXiv:1906.04721, 2019

Vineel Pratap, Awni Hannun, Qiantong Xu, Jeff Cai, Jacob Kahn, Gabriel Synnaeve, Vitaliy Liptchinsky, and Ronan Collobert. wav2letter++: The Fastest Open-source Speech Recognition System. arXiv:1812.07625, 2018

Sean I. Young, Wang Zhe, David Taubman, Bernd Girod. Transform Quantization for CNN Compression. arXiv:2009.01174, 2020

Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. arXiv preprint arxiv:1702.03044, abs/1702.03044, 2017.

Panayotov, Vassil, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur. "Librispeech: an asr corpus based on public domain audio books." In 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 5206-5210. IEEE, 2015.

Graves, Alex, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks." In Proceedings of the 23rd international conference on Machine learning, pp. 369-376. 2006.

Han, Song, Huizi Mao, and William J. Dally. "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding." arXiv preprint arXiv:1510.00149 (2015).