Background and Methods

This paper introduces a Deep Convolutional Neural Network called Scale-Invariant Temporal History Convolution Network (SITHcon) network. In this network, a CNN is built with a maxpool operation over a logarithmically-compressed temporal memory. The goal of this DNN is to provide a record or "memory" of the events in the recent past as a function of time at each moment. In this method, since the logarithmic compression of data is used, time scaling can be accommodated in the network by translation of data. This model is very closely realted to the DeepSITH network developed earlier. Both these networks use SITH as a representation of the past in compressed form. However, unlike the DeepSITH, the SITHcon model has a convolution layer and maxpool operation at each layer.

To evaluate the performance of this approach, the authors compare the results of the SITHcon network to Temporal Convolution Network (TCN). In TCN networks convolutions are made so that information leakage of future events don't happen in the past. This network has exponentially increasing dilations which provides each layer with (K-1) times the dilation.

The main distinction between these two networks is that TCN network applies convolutions directly on normal time whereas the SITHcon network applies it on compressed time. The effective history of TCN is limited by the number of layers and the dilation present in each layer.

Results

Experiment 1: Morse Adding Problem - The Morse Adding Problem is a task to evaluate the performance of the SITHCon and TCN networks on time series data. In this, the networks are presented with a two-dimensional time series input. The first dimension consists of a continuous stream of ten Morse code symbols, represented by numerical values from 0.0 to 0.9. The second dimension contains two activation pulses indicating which Morse code symbols are to be added. The goal is to decode the Morse code symbols indicated by the activation pulses and then perform addition. Once trained, their performance is evaluated by testing them on input sequences at different scales. As the testing scale deviates from training scale, TCN suffers loss in performance but SITHcon is still viable and doesn't show performance degradation. We can see in Figure 1 that the mean squared error in TCN grows very rapidly when the scale of data is changed whereas the MSE is almost constant and very low even when the scale is varied in SITHcon. It is to be noted, that while doing the experiment, I stopped training the SITHcon network once the model reached a threshold in loss which I chose to be very low. This optimization has been done to accommodate the compute resources available. The same threshold value was used to conduct the experiment for TCN network as well to maintain consistency and get the results.

Experiment 2: Morse Decoding Problem - The Morse Decoding Problem is a task in which the networks are trained to differentiate between the 43 different Morse code symbols presented as a one-dimensional time series. Each symbol in the morse code is depicted using a unique pattern of dashes and dots. In the experiment, the TCN and the SITHcon networks are trained on a single scale. After training, their performance is evaluated by testing them on morse code symbols presented at a range of unseen temporal scales. The goal is to check the network's ability to generalize to different timescales. In the results we notice that when the predictions are made for the same scale as the training scale, both the models have accuracy of 1. However, with varying timescales, we see that only SITHcon model can maintain the accuracy to 1 and the TCN model drops to almost zero. This can be seen in Figure 2.

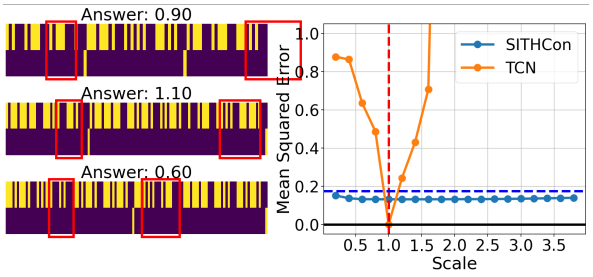

Figure 1: Morse Adding MSE Results

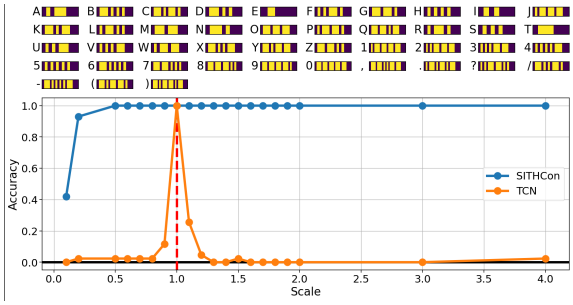

Figure 2: Morse Decoding Accuracy Results

General Discussion and Future Plans

In the next part of the project, I am planning to apply the SITHcon model and compare it with TCN model on a new dataset. The dataset which I have decided to explore is - Basic Arabic Vocal Emotions Dataset. This dataset is open-source and is available on Kaggle. (https://www.kaggle.com/datasets/a13x10/basic-arabic-vocal-emotions-dataset/data). This dataset contains 7 arabic words identified and named in Arabic language. It has 1935 records with 61 speakers. The sample rate is 16 kHz. I aim to test how the SITHcon model and TCN model produce results on this dataset when we give them different timescales as input.

Introduction

In the final phase of the project, I was inclined to implement this method and test its efficiency for a sound/audio dataset. After careful research, I chose the Basic Arabic Vocal Emotions Dataset. I try to solve a classification problem in this dataset. The audio files in this are spoken digits in arabic and a label(digit value) is assigned to each audio file. I have done the task of classifying audio files to digits represented by the audio. I have implemented scaling of the audio signals to [10x, 5x, 2.5x, 1.25x, 1x, .8x, .4x, .2x, .1x]. The crux of my experiment is the generalization performance of scaled audio samples of the original samples on SITHCON and TCN models. Theorectically, the SITHCON model should work well with the scaled data as well, without retraining it on different scaled datasets. TCN on the other hand is expected to see a dip in performance when tested with scaled data.

Methods

In this experiment I have downloaded the dataset from kaggle and uploaded the dataset to my google drive and I access that from colab environment.

After loading the dataset, I do scaling and feature extraction using the librosa library. I have extracted the spectrogram for each audio file. This spectrogram represents the audio signal in a numpy ndarray. After this, I calculate the short time fourier transform(stft) for each audio signal. After this, I shape using padding of zeros to make it conform with the shape of the SITHCON and TCN models for training. I also associate labels with each audio file. The labels are extracted from a part of a string from the audio file name.

Scaling is done by resampling each audio file by changing the sample rate of each audio file. The original sample rate is 16KHZ which is multiplied by the value in our scales array to get the new scaled audio signals. For each scaled audio signal, the spectrograms are obtained by the method discussed above. I have used the librosa library to do these tasks.

After this, I generate train test splits for each of the scales. This gives us trainX, testX, trainY and testY for each scale. I save theses files for each of the scales. This is a very important step as the train and test sets should be consistent when running on both SITHCON and TCN models to generate meaningful results for comparison.

I have also visualized some features of a random audio file from the dataset. Various features such as Mel-frequency cepstral coefficients(MFCC), Chroma features, Spectral contrast and Mel-Frequency spectrogram have been plotted.

I do the data generation when I train the SITHCON model and save data to be used in TCN model. I omit the data generation part when doing TCN training and evaluation. I just load the npy files created in data generation for SITHCON model.

I have trained both the models for 100 epochs with a train-scale of 10 and same hyperparameters to maintain consistency when comparing results. In the training, I have used the trainX, trainY, testX and testY of the scale 1 for training

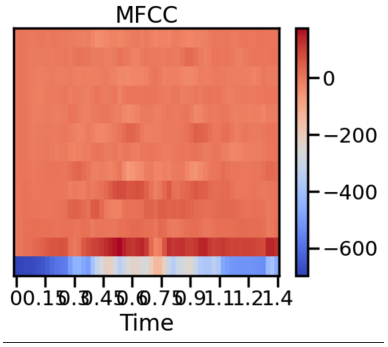

Figure 3: MFCC

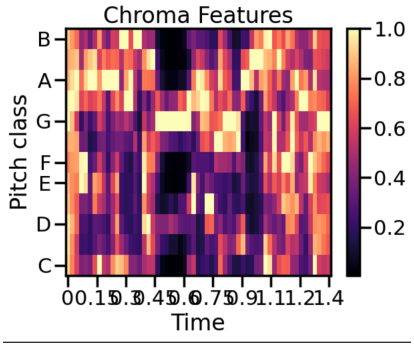

Figure 4: Chroma Features

both SITHCON and TCN models.

The code files I have used are uploaded in my BOX and the link is - (https://rutgers.box.com/s/fzqmgf3ssjsgtahp5uw0ytvyl9yz5nmd) The link for the dataset used from Kaggle is - (https://www.kaggle.com/datasets/a13x10/basic-arabic-vocal-emotions-dataset/data)

Results

After training, the models performance scores are measured. Both the models have a very close and similar performance scores when tested with data witch a scaling factor of 1(i.e original data). The effects of scaling are more evident in audio datasets as the process of scaling is very common practice as audio data can be sampled at different rates to achieve scaling.

As we can see from the graphs of the two models, the SITHCON model has considerably very good performance on the scaled data whereas the TCN model has very very poor performance when tested on a dataset which is different from what it was trained on. These observations again confirm with the findings and the results in the other textual data experiments that SITHCON has a very good generalization performance when tested with different scale of data than its training dataset.

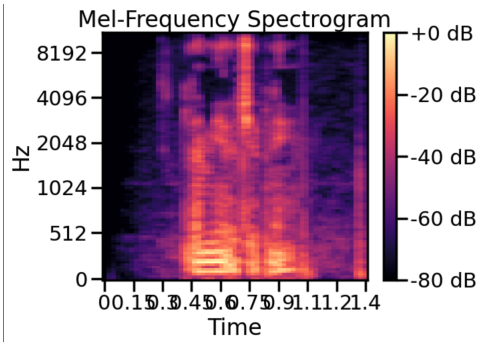

Figure 5: MEL Spectrogram

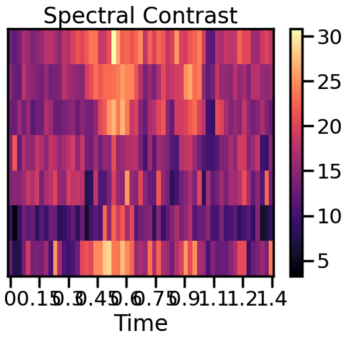

Figure 6: Spectral Contrast

Discussion

The SITHCON model is designed to be scale-invariant in time, from which I infer that it can handle changes in the scale of the input data without needing to be retrained. This is achieved by using a logarithmically-distributed temporal memory, where the peaks of the receptive fields form a geometric series such that the population codes a set of temporal basis functions over log time. A max-pool operation results in a network that is invariant to rescalings of time modulo edge effects. This feature allows the SITHCON model to generalize to different time scales, which is particularly beneficial in audio signal processing where the audio data can be sampled at different rates.

The SITHCON model has been shown to generalize without retraining to rescaled versions of the input, even when the input data is significantly rescaled. This property could lead to large-scale networks with dramatically different capabilities, including faster training and greater generalizability, even with significantly less hyperparameter tuning.

Some of the limitations which I experienced or thought about during my hands-on with this model involve computational complexity. The SITHCON model's architecture, which includes a logarithmically-distributed temporal memory, can be more computationally complex than simpler models like the TCN. This could potentially make the SITHCON model slower to train and more resource-intensive, especially for large datasets or complex models.

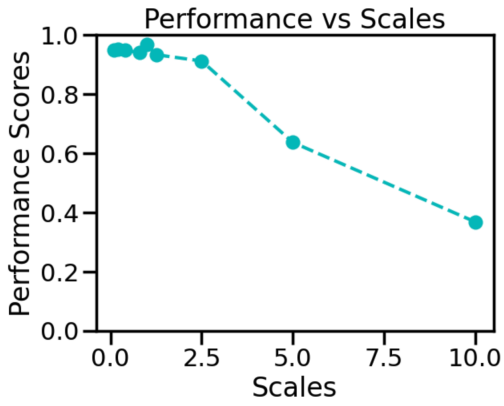

Figure 7: Sithcon Network Accuracy Results

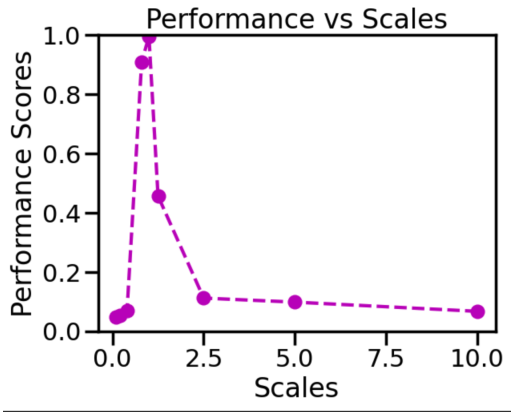

Figure 8: TCN Network Accuracy Results

The SITHCON model's unique architecture and its ability to generalize to different time scales without retraining can make it more difficult to interpret. This lack of transparency can be a drawback in situations where interpretability is important, such as in regulatory or auditing contexts.

Unlike some other models, there may not be many pretrained SITHCON models available for use. This could make it more difficult to get started with the SITHCON model, especially for beginners or those without access to significant computational resources.

However, I can say that these are potential drawbacks which may or may not be experienced. The unique feature of SITHCON to "memorize" and generalize to test sets of scaled data is very impressive and commendable after doing several experiments.

References

Aouf, A. (2020). *Basic arabic vocal emotions dataset.* Kaggle. Retrieved from https://www.kaggle.com/ds/345828 doi: 10.34740/KAGGLE/DS/345828

Driedger, J., & Müller, M. (2016). A review of time-scale modification of music signals. *Applied Sciences*, *6*(2), 57.

Huang, Q., & Hain, T. (2021). Improving audio anomalies recognition using temporal convolutional attention networks. In *Icassp 2021-2021 ieee international conference on acoustics, speech and signal processing (icassp)* (pp. 6473–6477).

Jacques, B. G., Tiganj, Z., Sarkar, A., Howard, M. W., & Sederberg, P. B. (2022). *A deep convolutional neural network that is invariant to time rescaling.*

Liu, H., Chen, K., Tian, Q., Wang, W., & Plumbley, M. D. (2023). Audiosr: Versatile audio super-resolution at scale. *arXiv preprint arXiv:2309.07314*.

MatthewDavies, E., & Böck, S. (2019). Temporal convolutional networks for musical audio beat tracking. In *2019 27th european signal processing conference (eusipco)* (pp. 1–5).