Architecture Analysis of Multitask Learning for Assessing Quality of Actions

Akhila Ballari
Georgia Institute of Technology
Atlanta, Georgia
aballari6@gatech.edu

Jaswanth Sai Pyneni
Georgia Institute of Technology
Atlanta, Georgia
jpyneni@gatech.edu

Nitya Tarakad
Georgia Institute of Technology
Atlanta, Georgia
ntarakad3@gatech.edu

Abstract

The importance of using neural networks to assess the quality of actions is rooted in the rapid growing nature of the field with its many applications such as in sports and medicine. The prevalence of this subject makes it important that we find the best performance for such tasks to leverage models for assisting or replacing human bias in assessing actions. We focus on supervised architectures applied to diving actions. The current state-of-the-art approach to this problem uses a multitask learning (MTL) approach with simple architecture. We show here that adding more complex mechanisms to the the current state-of-the-art architecture does not add much value in terms in performance in action quality assessment (AQA).

1. Introduction

Our project focuses on improving the performance in assessing the quality of particular actions, specifically improving upon the MTL approach for AQA. After reading about the initial implementation of MTL for action quality assessment [3], we determined that we can experiment with more complex architecture in both the upstream task of processing videos as well as in one of the downstream tasks–the caption generation. With these changes in the architecture, we hope to see better performance in overall action quality assessment.

1.1. Background

The current work on using MTL for AQA is built from the original AQA work of Pirsiavash *et al*. [4] which focused on using pose features to generate an action quality score. Paritosh *et al*. [3] devised the C3D-AVG-MTL network to establish that an MTL approach of describing and commenting on actions can improve the AQA task. However, their approach relied on using simple mechanisms within the network, creating a space for possible improvement.

1.2. Motivation

Developing well performing models to assess action quality will enable more standardized performance standards during high profile competitions. Human judges will face human error that may arise in missing a certain action in a performance which could lead to bias in evaluating performance. Parsing through video footage of performances can be extremely long and requires as much human attention as watching live. Well trained models that can correctly analyze and assess performance in sports competitions will evenly distribute scores, eliminate human bias towards competitors, and greatly save on time during competitions. This will be very valuable to sports judges and commentators. While we focus on action quality assessment from diving tournaments, our work could be extended to other areas in AQA.

2. Related Work

A few research papers built on top of each other regarding assessing quality of actions are foundational to our project. Here we discuss some of these concepts from these papers and other architectures used to modify the MTL approach that are relevant to understanding our work.

2.1. Single Label AQA

Pirsiavash *et al*. [4] utilized Discrete Cosine Transformation (DCT) to create features from input frames of the body pose. These features were used for a support vector regression (SVR) model to generate an AQA score, using a single-label approach for modeling. Their work also introduced the original dataset for this problem space: frames from Olympic footage of Diving and Figure skating events that come with annotations from Olympic judges.

2.2. C3D Neural Networks AQA

Parmar and Morris [2] built on the work of Pirsiavash *et al*. [4] by utilizing 3D convolutional neural networks (C3D) for AQA. C3D is used to capture temporal representations of salient motion in the AQA task.

2.3. Multitask Learning Approach

Paritosh *et al*. [3] further expanded on [2] to devise the C3D-AVG-MTL network for the AQA task. In this network, Paritosh *et al*. introduced the concept of multitask learning to AQA by hypothesizing that creating a model to learn to describe and comment on an action in a video in parallel can help the model have better performance for the main task of AQA. Thus, the C3D-AVG-MTL network, as shown in Figure 1, extracts spatio-temporal features using a C3D backbone for upstream processing of input video frames, and then optimizes for AQA by aggregating three downstream tasks with individual loss functions. The downstream tasks are AQA scoring, action classification and captioning of video frames. Paritosh *et al*. [3] released the largest dataset for this problem by aggregating 1412 samples of diving action sequences from 16 different tournaments, to include samples from all genders, multiple views, and of varying levels of competitions. This dataset includes descriptions of the action from television broadcast, AQA score from judges' scores at the competition, and breakdown of the action (diving) into individual components (position, rotation type, etc..).

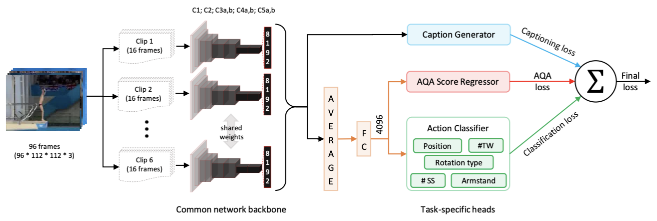


Figure 1: Original C3D-AVG-MTL network from Paritosh *et al*. [3]

2.4. Separable C3D Neural Networks AQA

In [6], Sun *et al*. describe VideoBERT, a SoTA text-to-video generation model that is built by using S3D [8]. S3D is built from adding separable temporal convolutions to I3D [8], which is a 3D Inception network [7] that convolves over space and time. S3D is less computationally expensive than C3D and has better accuracy in classification tasks than I3D [8].

3. Approach

We wanted to see if we could improve the performance of the overall action quality assessment through a series of architecture changes. The original architecture [3] consisted of a C3D backbone for upstream preprocessing and a GRU layer within one of the downstream tasks of caption generation. We utilized this architecture, shown in Figure 1, as our baseline for comparison with the rest of our modified architectures.

3.1. Changes in Backbone Structure

Our modified architectures consisted of one of two backbone structures. The first backbone tested is the C3D backbone originally described in Paritosh *et al*. [3]. This is the original backbone in the architecture for the MTL approach. The second backbone tested is the S3D [8] backbone utilized by VideoBERT [6]. As noted above, S3D is another spatio-temporal structure that separates computation over space and time, taking advantage of the order that the video frames occur in. Because S3D has been shown to have better performance than C3D and I3D [8] in classification tasks, we hope that this backbone will give us better results for the MTL-AQA task.

In order to modify the network to use the S3D backbone, we took an implementation of the model from [1], along with pre-trained weights and replaced the C3D backbone. We modified the S3D backbone to make sure the output dimensions of the new backbone match the expected inputs to the downstream tasks.

3.2. Changes in Caption Generation

As the generated captions are used a feature to the action scoring task, we modified the caption generation task by changing the RNN implementation. We hypothesize that better captioning will lead to a better representation of the action sequences which then will lead to more accurate action quality scores. The original RNN architecture utilized 2 stacked GRU cells [3]. We experimented with an LSTM implementation of the RNN as LSTMs generally result in better captioning as they have higher expressiveness. LSTM is another popular RNN cell type that utilizes 3 gates (input, output and forget) instead of only 2 gates (reset and update) as used by a GRU cell. We expect that a more complex RNN cell type will lead to better captioning. We tested to see if more layers (stacked GRUs or LSTMs) bettered the architecture's performance.

While experimenting with a change in cell type, we also experimented with using attention with the RNN layers. Attention is a mechanism that analyzes inputs based on asso-

Modified Architectures					
Experiment	Backbone	RNN Cell Type	Num of Stacked RNN cells	Attention Used	Feature Encoding
Baseline [3]	C3D	GRU	2	No	Averaging
1	C3D	GRU	2	Yes	Averaging
2	S3D	GRU	2	No	Averaging
3	S3D	GRU	2	Yes	Averaging
4	C3D	GRU	8	Yes	Averaging
5	C3D	LSTM	8	Yes	Averaging
6	C3D	LSTM	2	Yes	Averaging
7	C3D	GRU	2	Yes	LSTM

Table 1: Architecture Changes per Experiment

ciated context. Attention was implemented as a linear layer that takes the video features as input. The output of the attention layer is then passed into the RNN (either GRU or LSTM) used for caption generation. Because an associated context is utilized, we expect captioning and over all scoring to improve with the use of this attention mechanism.

With changes in RNN cell types for caption generation, we believe that caption will be better generated, allowing for better results in our downstream tasks and as a result, improve AQA.

3.3. Changes in Feature Encoding

After the frame features have been extracted from the clips, they are encoded before being passed into the fully connected layer in order to be used in the downstream tasks. The original approach in MTL [3] was to average the features over all the clips for every video that were extracted before passing into the fully connected layers 1. Because averaging disregards the temporal aspect of the frames extracted, we decided to also experiment with passing the extracted features into an LSTM encoder instead of averaging them. Using an LSTM encoder would uphold the temporal aspect of the frames as they are passed through the fully connected layers and into the downstream tasks, overall showing better results.

To implement the LSTM encoder, we replaced the averaging across clips with an encoder [5]. We passed the extracted clip features into the built encoder. The encoded features were then passed into the fully connected layers.

3.4. Modified Architectures

Table 1 lists the architecture changes for each experiment. As described in previous sections, the tested architectures have changed in backbone (C3D vs S3D), the RNN cell type (GRU vs LSTM), the number of stacked RNN cells, use of attention on the captioning side, and the type of feature encoding (averaging vs an LSTM encoder).

3.5. Anticipated and Encountered Experiment Issues

The only issue we anticipated was a memory error when replacing the averaging of clip features done in the upstream task with an LSTM encoding. We expected this error because of the sheer amount of clip features that need to be encoded.

While running our experiments based on our approach, we experienced a couple of issues along with our anticipated one. We had an issue with the data we were utilizing. We had to cut down our training and test data sizes because using all of the data at our disposable caused a longer running time for our experiments. In order to run all of our experiments in time to compare, we ended up cutting down the data. Another issue we came across for data preprocessing was extracting the frames from the Youtube videos provided. We noticed that there was no explanation for how frames were extracted, so we began to write our own script to parse the videos. After getting in contact with Paritosh Parmar regarding the specifics of frame extraction, he provided the frame extraction script he used for his MTL-AQA research.

4. Experiments and Results

4.1. Measuring Success + Loss

Our goal was to improve performance by reducing training loss and increasing testing correlation by the 100^{th} epoch. The training loss, L_{MTL} (1) is a summation of the losses from the three downstream tasks: Action Quality Assessment, Action Recognition, and Captioning Tasks.

As defined in [3], the action quality assessment loss L_{AQA} (2) is a summation of the L1 and L2 losses between the predicted score x_i and the ground truth score y_i for each of the N samples. The action recognition loss L_{Cls} (3) is a cross entropy loss between the predicted labels $y_{i,j}$ and ground truth labels $x_{i,j}$ summed over the categories in each sub-action class sa. The captioning loss L_{Cap} (4) is a negative log likelihood loss where sl is the sentence length. Therefore the overall loss function to be minimized is defined as the weighted sum of these losses.

$$L_{MTL} = (1)L_{AQA} + (1)L_{Cls} + (0.01)L_{Cap} \quad (1)$$

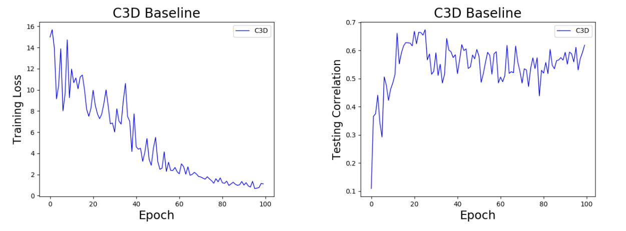
$$L_{AQA} = \frac{-1}{N}\sum_{i=1}^{N}(x_i - y_i)^2 + |x_i - y_i| \quad (2)$$

$$L_{Cls} = \frac{-1}{N}\sum_{i=1}^{N}\sum_{sa}\sum_{j=1}^{k_{sa}} y_{i,j}^{sa}\log(x_{i,j}^{sa}) \quad (3)$$

$$L_{Cap} = \frac{-1}{N}\sum_{i=1}^{N}\sum_{sl} y\ln(x^{cap}y^{cap}) \quad (4)$$

Testing correlation is a measure of the statistical relationship between the predicted AQA score for a dive and the ground truth score as provided in the data. A higher correlation means a better performance in the model, with the predictions being closer to the annotated scores.

4.2. Baseline



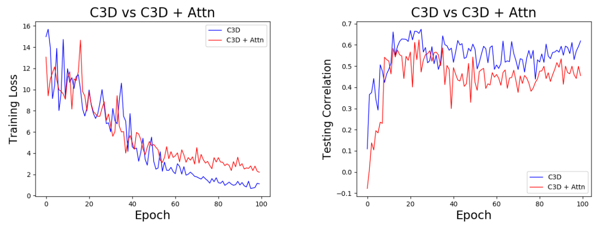
(a) Loss (b) Correlation

Figure 2: Baseline

Figure 2 shows the training loss and testing correlations for the baseline architecture that was implemented in [3]. The baseline architecture is evaluated on a subset of the dataset [3]. This baseline was trained with 100 epochs, saving and evaluating every epoch. It used an Adam optimizer with a 0.0001 learning rate. The dataset used for reproducing the baseline and running the experiments is a subset of the dataset used in [3]. This dataset is described in detail in the Appendix. Since we test this baseline architecture with a very small subset of the data, we see a testing correlation of about 60% (the original MTL paper states 90.44% testing correlation on the entire dataset [3]). The following experiments are in comparison to this reproduced baseline.

4.3. Experiment 1: Adding Attention to Baseline

In the baseline, the video features are directly passed into a RNN. To test how better captioning will affect our performance, we added an attention layer to the captioning downstream task. The below figures display the results of adding attention to the baseline.



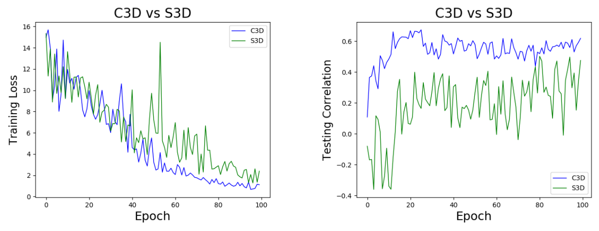
(a) Loss (b) Correlation

Figure 3: Experiment 1

Surprisingly, attention did not improve the overall performance. Attention resulted in a higher training loss and lower testing correlation. We would expect that attention would perform bad if there was overfitting, but the graphs indicate that this is not the case. While attention performs worse than the baseline, the performance does not suffer too much. Testing other kinds of attention mechanisms would be a good topic for further research.

4.4. Experiment 2: Replacing C3D Backbone with S3D Backbone

S3D replaced the C3D backbone of shared weights that extracts features from each clip from the baseline. The following graphs display the results of replacing C3D with S3D.



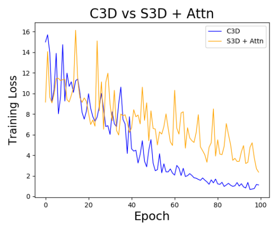
(a) Loss (b) Correlation

Figure 4: Experiment 2

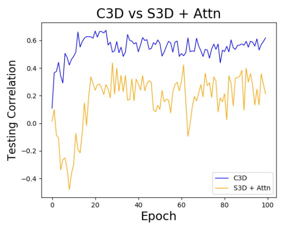
S3D resulted in slightly higher training loss and significantly lower testing correlation. This is a case of underfitting the model to the dataset. This is reasonable since S3D is a simpler architecture that separates convolutions over space and time than that of C3D. Even though VideoBert uses S3D to extract and represent features from video clips, using S3D alone is not sufficient to model this problem.

4.5. Experiment 3: Replacing C3D Backbone with S3D Backbone and Adding Attention

The following figures show our results from testing attention with the S3D Backbone.



(a) Loss



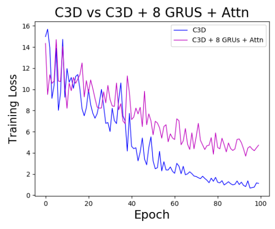
(b) Correlation

Figure 5: Experiment 3

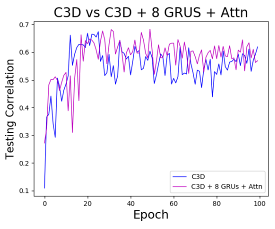
As our results from Experiment 1 and 2 were not promising, it is reasonable that combining the S3D and Attention changes to the architecture would not be promising. As these graphs more closely resemble the graphs from experiment 2, it shows that the backbone architecture has a higher impact on overall performance than adding attention in the captioning downstream task. Having run these experiment simultaneously, we could have pivoted earlier had we know the results from Experiment 1 and 2 sooner.

4.6. Experiment 4: Stacking GRUs with Attention to Baseline

The original baseline stacks 2 GRUs in the RNN implementation. We increased the number of stacked GRUs to 8. As we expected attention to improve captioning, we also added the attention layer to this experiment. The following graphs show the results from stacking GRUs with Attention in the baseline architecture.



(a) Loss



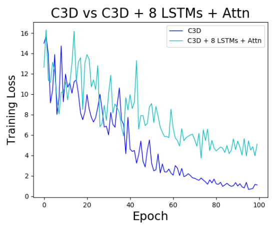
(b) Correlation

Figure 6: Experiment 4

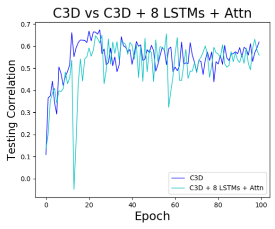
The training loss is higher with the increased number of stacked GRUs, but the testing correlation is slightly higher on average than the baseline. Even though the testing correlation looks promising, the performance improvement is very borderline. This experiment makes it look like the original baseline architecture might be slightly overfitting the data. A good follow up experiment would be to test on the entire dataset to evaluate how stacking more RNN cells would affect performance.

4.7. Experiment 5: Stacking LSTMs with Attention to Baseline

Since the original architecture utilizes 2 GRUs in the RNN implementation, we wanted to test how a more complex RNN implementation with stacked LSTMs would affect our results.



(a) Loss



(b) Correlation

Figure 7: Experiment 5

The training loss is similar to that of experiment 4. The testing correlation is on average about the same as that of the baseline. The analysis is the same as that in Experiment 4. However, since GRUs have proven to perform better than LSTMS, future work would be based on the GRU implementation. Similar to how attention does not lead to better performance, the higher expressiveness of LSTMs does not add much value to the overall performance.

4.8. Other Results

Experiment 6 was stopped early since plotting the losses and correlations showed that LSTM with Attention did not perform better than the baseline. To save on time and resources, we decided to run the other experiments that we expected to perform better. We also do not have results for Experiment 7 due to CUDA memory issues. Our setup on GCP did not support this experiment. We further tried to test this experiment on a GT lab cluster with multiple GPUs and with a smaller batch size, but still faced the issue. If given sufficient resources, we expect this experiment to lead to significant performance improvements.

The following figures show the training losses and testing correlation in comparison to each of the experiments.

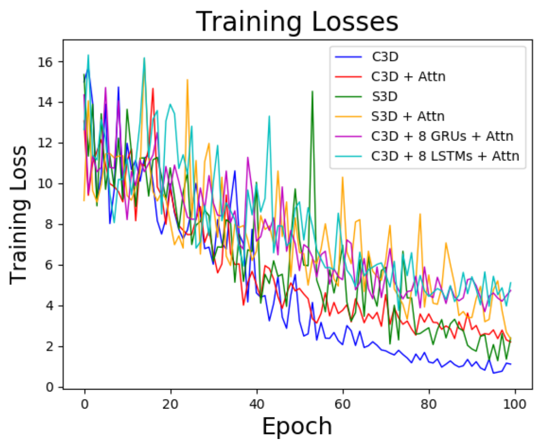


Figure 8: Overall Training Losses

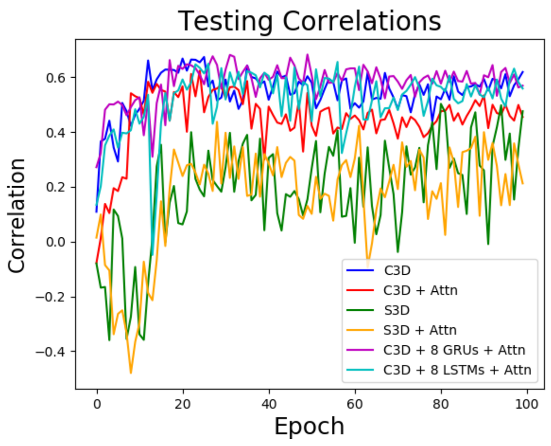


Figure 9: Overall Testing Correlations

5. Discussion and Next Steps

Our results show that we did not achieve significant improvements in the overall action quality assessment. Our assumptions on the impact of using S3D proved to be the worst performing. Even though the MTL architecture is quite simple, it performs significantly well. While we were able to reach a slightly higher correlation by stacking more GRU cells, this performance benefit was not substantial. It would be interested to test the maximum performance increase we can achieve with the largest number of stacked GRUs.

Looking into future research, one next step would be to reproduce the LSTM encoder experiment using a higher amount of compute. As mentioned earlier, we believe that averaging the extracted features is a naive approach because it doesn't take into account the spatio-temporal aspect of the features. Therefore, utilizing an LSTM encoder on the features could potentially help improve the performance of the model. It would be worthwhile to train and test the best performing experiments with the larger dataset. Other future directions include applying this system to other sports to see how well the architecture generalizes.

References

[1] Kyle Min and Jason J Corso. Tased-net: Temporally-aggregating spatial encoder-decoder network for video saliency detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2394–2403, 2019. 2

[2] Paritosh Parmar and Brendan Tran Morris. Learning to score olympic events. *CoRR*, abs/1611.05125, 2016. 2

[3] Paritosh Parmar and Brendan Tran Morris. What and how well you performed? a multitask learning approach to action quality assessment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 304–313, 2019. 1, 2, 3, 4, 7

[4] Hamed Pirsiavash, Antonio Torralba, and Carl Vondrick. Assessing the quality of actions. 10 2014. 1, 2

[5] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. 2016. 3, 7

[6] Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. *CoRR*, abs/1904.01766, 2019. 2

[7] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. 2

[8] Saining Xie, Chen Sun, Jonathan Huang, Zhuowen Tu, and Kevin Murphy. Rethinking spatiotemporal feature learning for video understanding. *CoRR*, abs/1712.04851, 2017. 2, 7

A. Appendices

A.1. Experiment Details

A.1.1 Models and Algorithms

C3D is used as the backbone in the orignal MTL approach model. A C3D model stands for ”Convolutional 3D”, which is comprised of 3D convolutional layers. More information about the usage of C3D Neural Nets can be found in Related Work.

S3D is used in the framework for VideoBERT. S3D stands for ”Separable 3D CNN,” in which the network convolves over space and time. It is a spatio-temporal structure built off of Inception networks. More information about the usage of S3D Neural Nets can be found in Related Work.

RNN RNN is a recurrent neural network. These networks contain hidden states that that allow previous outputs to be used as inputs. This allows us to model temporally distinct data. Gated Recurrent Unit, GRU, is a type of RNN cell that utilizes 2 gates: reset and update. Long Short Term Memory, LSTM is a type of RNN cell that utilizes 3 gates: input, output and forget.

LSTM Encoder is used as a replacement to the original feature encoding done in the MTL approach [3]. The original MTL model averages over the extracted features from the frames collection in order to encode them. One of our experiments attempted to utilize an LSTM encoder instead to take advantage of the spatio-temporal aspect of the frames.

A.1.2 Performance Metrics

Testing and training measures can be be found in section 4.1.

A.1.3 Datasets

The dataset we used to train and test our baseline and modified architectures is the same data used in the MTL approach [3]. The full dataset has 1412 diving samples, collected from 16 diving tournaments. In order to cut down the training and testing time, we used a 140 sample subset for training and a 50 sample subset for testing. Information on the data and how to get the subsamples we used is mentioned in our Github repository (see A.1.3).

A.1.4 Source Code

Our source code can be found here: `https://github.com/jpyneni3/MTL-AQA`. In addition to building off of the MTL approach in the repo made by Paritosh *et al.* [3], we also utilized code from a repo for the S3D model [8] and code for the LSTM autoencoder [5]. The code repository for the S3D model can be found at `https://github.com/kylemin/S3D` and the code repository for the LSTM encoder is at `https://github.com/shuuchen/video_autoencoder`.

A.1.5 Compute Infrastructure

We used Google Cloud Platform’s Deep Learning VM with a NVIDIA Tesla K80 GPU. In order to run models in parallel for faster experimentation, we set up 2 identical VMs with the necessary data. See Setup for more info.

A.1.6 Runtime

With this setup, it takes 100 hours to train the baseline model on the entire data. Using the subsample dataset, each model took an average of 24 hours for training and testing.

A.1.7 Experimental Results

We trained over the all the data in the subsample for 100 epochs and tested all the data each epoch. Our results for training compare the training loss and the results for testing compare the training correlation. For each experiment run, we compared the training loss and testing correlation progression to those of the baseline test, the original framework used in the MTL Approach [3]. See Experiments and Results for more info. Further clarifications and requirements to run our experiments can be found in the README.