Self-Supervised Machine Listening: Fall 2019 Capstone Report

B V Nithish Addepalli *
NYU Center for Data Science
bva212@nyu.edu

Jatin Khilnani *
NYU Center for Data Science
jk6373@nyu.edu

Ravi Choudhary *
NYU Center for Data Science
rc3620@nyu.edu

Shreyas Chandrakaladharan *
NYU Center for Data Science
sc6957@nyu.edu

Vincent Lostanlen
Music and Audio Research Lab
NYU Center for Urban Science and Progress
vincent.lostanlen@nyu.edu

Brian McFee
Music and Audio Research Lab
NYU Center for Data Science
brian.mcfee@nyu.edu

Abstract

In recent decades, supervised learning using deep neural networks has shown tremendous results in Computer Vision, Natural Language Processing and several other fields. A major drawback of these networks is their necessity for a large amount of labelled examples to learn from. Labeling a large number of examples requires a lot of effort and domain specific expertise, especially for audio related tasks like instrument detection in music. Recently, a new paradigm of representation learning called Self-Supervised Learning has emerged where useful representations are learnt from data without the necessity of human annotated labels. These representations greatly reduce the amount of labelled data required to solve any supervised task. In this project, we investigate the effect of Self-Supervised Learning in solving tasks in the domain of Machine Listening. To this end, we start with designing a Convolutional Neural Network to solve the task of instrument detection. Consequently, we study the effect of two self-supervision techniques - Time Reversal and Jigsaw in reducing the amount of labeled data required to solve the task. We show that we can improve the performance on the task using self-supervision, subject to the conditions of our experiment as elaborated in the paper. We also contribute a novel feature extractor to the domain of Music Information Retrieval which can detect whether a music clip is sequenced correctly.

*Equal contribution

1 Motivation

The scientific field of machine listening is the auditory equivalent of computer vision. It aims at analyzing and transforming acoustic data in ways that are informative to humans. Three well-known industrial applications of machine listening are automatic speech recognition (e.g., Amazon's "Echo"), music recommendation (e.g., Spotify's "Made For You"), and audio fingerprinting (e.g. Apple's "Shazam"). Over the past decade, this field has witnessed a breakthrough of new algorithms, known as convolutional neural networks (convnets). convnets operate by scanning the time-frequency domain and producing a heatmap of relevant patterns. Empirical evidence demonstrates that organizing convnets as a deep cascade of non-linear transformations is beneficial to classification accuracy.

Nevertheless, increasing the depth of a convnets also increases the need for human-labeled training data. Although the topics of image recognition and speech recognition both enjoy vast amounts of such data, the same cannot be said of music nor bioacoustics. This is because identifying the full instrumentation of a musical piece, or identifying the species composition of a field recording, demands a high level of domain specific expertise, thereby raising the cost of human annotation. We want to investigate the effect of self-supervision in alleviating the need for human intervention in the design of convnets for machine listening.

2 Self-Supervised Learning

Deep Neural Networks have revolutionized the fields of Computer Vision and Natural Language Processing. However, their performance directly depends on the amount of labeled data they are trained on. Finding large curated datasets with appropriate labels is hard for most problems and creating such datasets from scratch is harder. Labeling data is an expensive and laborious process. To overcome the necessity of large annotated dataset, a new paradigm called Self-Supervised Learning (SSL) has emerged. SSL has recently gained a lot of traction in the field of Machine Learning [1]. In this paradigm, deep neural networks are first trained to predict some structural aspects of the data. This task is called a 'pretext' task and learning to solve this task helps the network learn meaningful representations of the data. These representations can then be used to solve any related task more efficiently in terms of the amount of labeled data required and the speed of convergence to optimal performance. The related task to be solved is called as the 'downstream' task. In a sense, SSL exploits the innate structure present in data to learn patterns about the data and thereby requires less supervision while solving any objective task with the data.

A famous domain where SSL has been used to great success is Image Recognition. Some effective pretext tasks in image recognition are: distinguishing natural images from images that are upside down [2]; distinguishing natural videos from videos that are played backwards [3]; and learning to solve the jigsaw of an image [4]. Because these geometric perturbations of the data can easily be automated, formulating pattern recognition in terms of SSL allows to tap into a virtually infinite amount of training data. Another domain demonstrating the effectiveness of SSL is Natural Language Processing — wherein BERT [5], a Transformer model [6] trained on pretext tasks of Masked Language Modeling and Next Sentence Prediction has given state-of-the-art results on several benchmarks.

Although SSL is quickly gaining momentum, there are, as of today, few published applications of SSL on Machine Listening. The goal of this project is to contribute to filling this gap in knowledge. In particular, we will devise a judicious pretext task for machine listening; use it to train a convnet on large volumes of audio data; and re-use this convnet for fine-grained classification experiments, such as musical instrument recognition and bird species identification.

3 Related Work

Majority of the research in the domain of machine listening with self-supervision has used multi-modal supervision to achieve state-of-the-art results, viz. [7] [8] [9] [10] [11] [12]. Two of the seminal papers that we study are [11] and [12]. Both of these papers use Audio - Visual Correspondence as cross-modal self-supervision to generate audio embeddings and visual embeddings. These embeddings are used in downstream tasks such as video classification, video tagging and object localization. The usage of SSL improves the performance and decreases the data dependency in both cases. A key insight from these papers is the tendency of self-supervision to use undesirable shortcuts and exploit patterns in the data to 'cheat'. We use counter strategies to prevent the same, which are elaborated in Section 5 under Data Preparation.

An alternate line of research in the same domain is undertaken in [13]. It defines a novel pretext based on predicting the temporal structure of data. Using this temporal structure eliminates the need for an additional modality (e.g. images or videos). Nevertheless, it requires labels to indicate the temporal structure. For example, they utilize timestamp of each data point in the available sensor network data.

Our work is majorly inspired from SSL and associated pretext tasks used in Computer Vision [4][3]. [4] introduces a pretext task to identify the ordering of scrambled pieces of an image, much like solving a jigsaw puzzle. This pretext task was effective in learning representations of images and beat several benchmarks in image recognition and transfer learning. Additionally, this pretext task is purely unsupervised in the sense that it requires no labels or extra modalities. While [3] uses videos to train a classifier to predict if the video is playing in the natural order or whether it has been reversed. The classifier is trained to use the cues based on motion of objects and concepts of physics like gravity to detect the direction of flow of time.

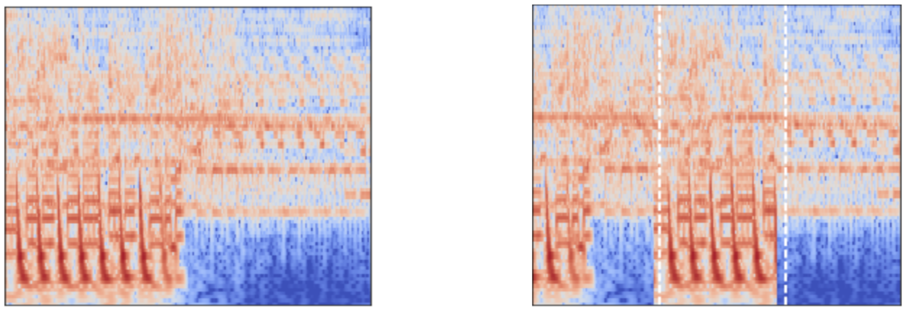
We propose to use pretext tasks similar to [4][3] and test their effectiveness in learning audio representations. Our approach to self-supervision in machine listening is markedly different from approaches in the domain so far. Our approach is purely unsupervised and does not require temporal labeling like [13] or visual modality like [7] [8] [9] [10] [11] [12].

4 Problem Definition

4.1 Pretext Tasks

We define two pretext tasks that try to predict the spatial structure of audio clip spectrograms. Spectrograms are visual representations of audio in the time-frequency domain. Details about how we convert audio clips into spectrogram images can be found in Section 5 under Data Preparation.

Our first pretext task is called **Jigsaw**. In this task, we split the spectrogram of an audio clip into three equal parts along the time axis. We shuffle these pieces randomly and train a classifier to predict whether the spectrogram is shuffled or not. A music clip is usually composed with repeating beats. If we jigsaw a musical piece then its beats will be out of order in most of the cases. Therefore, we believe that the network has to learn intrinsically about a musical piece to classify whether it has been jigsawed. Refer Figure 1 for an example of a spectrogram that has been jigsawed.

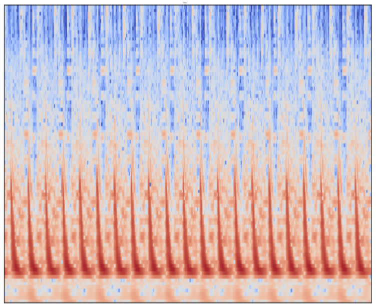

(a) Original Spectrogram (Order: 1,2,3)

(b) Jigsawed Spectrogram (Order: 2,1,3)

Figure 1: Example of Jigsaw Pretext task

Our second pretext task is called **Time Reversal**. In this task, we flip the spectrogram of an audio clip along the time axis. We then train a classifier to predict whether the spectrogram is flipped or not. If the sound is naturally generated, it releases vibrations and dissipates energy which in turn increases entropy. Playing in a audio file in reverse can show the reverse phenomena which doesn't occur naturally. Refer Figure 2 for an example of a spectrogram that has been reversed along the time axis.

Solving both of these pretext tasks requires the classifier to understand the spatial structure of spectrograms and reason about the organization of any given spectrogram. This suggests that the classifier will learn useful representations of spectrograms to help it solve the task. Our hypothesis is that these representations will be useful in solving a variety of downstream tasks with less labeled data.


(a) Original Spectrogram

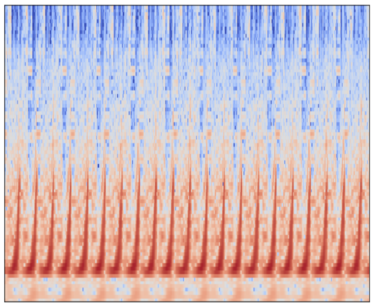

(b) Reversed Spectrogram

Figure 2: Example of Time Reversal Pretext task

4.2 Downstream Task

We choose instrument recognition as our downstream task. Instrument recognition is a long standing and challenging problem in machine listening. The task is to build a classifier that detects the presence of each instrument class in an audio excerpt. We will establish a baseline performance on this dataset by training a randomly initialized classifier and compare the baseline performance with a classifier using the representations learned from the pretext task.

5 Data

5.1 Data Source

For the pretext task, we use the **Free Music Archive (FMA)** [14] dataset. The Free Music Archive (FMA) is a web-based repository of freely available music recordings. Recently, a snapshot of FMA has been released to the research community to facilitate content-based music analysis evaluation. The FMA snapshot includes 106,574 tracks of 30 seconds each by some 16,341 artists, along with pre-computed features which is referred as large version. There's also a subset of this dataset, referred as small version, which has 8,000 tracks. We train models on both small and large versions of this dataset as our source of unlabeled data for learning audio representations and present our insights for the pretext tasks.

The dataset we use for the downstream task is **OpenMIC-2018** [15]. This dataset contains 20,000 examples of Creative Commons-licensed music available on the Free Music Archive. Each example is a 10-second excerpt which has been partially labeled for the presence or absence of 20 instrument classes by annotators on a crowd-sourcing platform. The data was made in a way to ensure that each class has at the least 500 positively tagged and at least 1500 positive or negative tagged examples.

5.2 Data Preparation

5.2.1 Pre-processing

The entire size of our pretext task dataset (FMA) is 93 GB. It consists of 106k mp3 files each of which is an independent datapoint. Considering we work with spectrograms instead of the raw mp3 file, we convert all the 106k mp3 files into spectrograms and store it. This pre-processing step avoids having to convert each time we load a data point. It reduces the time taken to retrieve one batch of size 32 from around 120s to 0.2s. Also, we used Python's multiprocessing library to pre-process the audio files in parallel.

We do the same pre-processing step for our downstream task dataset (OpenMIC) which contains 20k mp3 files.

We use the *librosa* library for processing the mp3 files. We load each mp3 file and convert it into an amplitude spectrogram using the Constant-Q Transform, which is similar to Fourier Transforms and extracts the power spectrum from waveforms. The Constant-Q transform converts the frequency scale into log-frequency scale which is more invariant to translation and hence more suitable for using with 2-D convolutions. [16]. We use the absolute value of the amplitude of the wave and discard the phase information. Next, we convert the amplitude spectrogram into a db-Scaled spectrogram, the decibel scale is more perceptible for humans.

5.2.2 Pretext Data Preparation

The 20k tracks in the OpenMic, our downstream dataset, are a subset of the tracks present in FMA. We exclude these 20k overlapping tracks from the training data for pretext tasks to avoid data leakage. We use a 0.8 train validation split ratio to obtain our train and validation splits. For each audio file, we choose a k second contiguous sub-part by sampling a starting point at random and clipping the file after k seconds from that point. A mathematical convenience is that the number of seconds in the raw waveform is directly proportional to the length of the spectrogram. So we can clip the audio files on-the-fly even after converting to spectrograms.

For the time reversal pretext task, we randomly flip each k second spectrogram along the time axis with the probability of 0.5. Again, a mathematical convenience is that flipping a spectrogram is equivalent to flipping the raw waveform. Each datapoint is of the form (x,y) where x is the spectrogram of k second audio clip and y is a binary label indicating whether the spectrogram has been flipped with 1 indicating flipped spectrogram.

For the jigsaw pretext task, we split each spectrogram image into three pieces and shuffle them randomly. By shuffling, we mean randomly choosing a permutation for the three pieces out of a total six permutations, each with equal probability of being chosen. Each datapoint is of the form (x,y) where x is the spectrogram of k second audio clip and y is a binary label indicating whether the spectrogram has been shuffled with 1 indicating shuffled spectrogram.

As found in [4], neural networks try to cheat in solving the task by using the pixel information at the boundaries, edge continuity, of the pieces instead of learning useful representations. To avoid these trivial shortcuts, we removed 5 pixels from the edges of the each of the jigsaw pieces.

5.2.3 Downstream

A key characteristic of the tracks in OpenMic dataset is that they can be related to each other. Though the dataset has 20k tracks, not all pairs of tracks are independent. Some of those tracks can belong to the same musical performance i.e recorded at different points of time during the performance. Considering this, if we randomly split the dataset into training and testing data, we might induce data leakage. Conveniently, the dataset has an official train and test split to prevent any leakage due to this issue. Likewise in our project, the official test set from OpenMic dataset has been used to measure the downstream performance. The official training set was divided into training and validation set to be used for early stopping with a train-validation split ratio of 0.8.

Each data point in OpenMic has a music file, a masking array and a label array. The masking array indicates the instruments for which the music file was annotated. The label array gives the confidence of the labels on the presence or absence of the instrument. A threshold of 0.5 is used on the label array to distinguish between the presence and absence of the instruments. We massage each downstream data point depending on the architecture adopted for solving the pretext task. For the convolution only architecture, we pass the whole spectrogram of the data point. For the convolution with fully connected layer and the ResNet18 architecture, we divided the spectrogram into 3 equal length parts, like in the Jigsaw pretext task, and pass these 3 parts as separate channels in ResNet18 and as separate inputs in the convolution with fully connected layer architecture. We also normalize the data using sample mean and variance calculated for the whole training data.

6 Experiment Setup

6.1 Model Architecture

Pretext Architecture

Most of our architecture decisions are inspired by [11] and [12]. We experimented with 3 different convolution architectures.

- **ConvNet**
  ConvNet is our baseline architecture [3]. It is inspired from [12]. It is a seven layer convnet with 3 5x5 kernels, 2 4x4 kernels and 1 3x3 kernel. Every conv layer is followed by a BatchNorm and ReLU activation. It has a final 2-way classification layer to output the probabilities of each class.
- **ConvNet-ext**
  During our experimentation, for the Jigsaw task, we saw that **ConvNet** wasn't able to perform well on the task, and we surmised that the task required more parameters. So, we experimented with

a second architecture **ConvNet-ext** [3]. **ConvNet-ext** is an extension of **ConvNet** by inserting 2 fully connected layers after the seven layer convnet followed by a final 2-way classification layer.

- **ResNet18**
  We also experiment with ResNet18 [17] to test our hypothesis, the effectiveness of self-supervision, with a proven architecture in the field of computer vision. We adapt the ResNet18 architecture by swapping the final classification layer of ResNet18 with a 2 way classification layer as suited for our problem.

Downstream Architecture

For the downstream task, we test all the 3 pretext model architectures. The only change in the architecture is swapping the final 2-way classification layer with a new 20-way classification layer.

Refer Figure 3 for a visual overview of our model architecture.

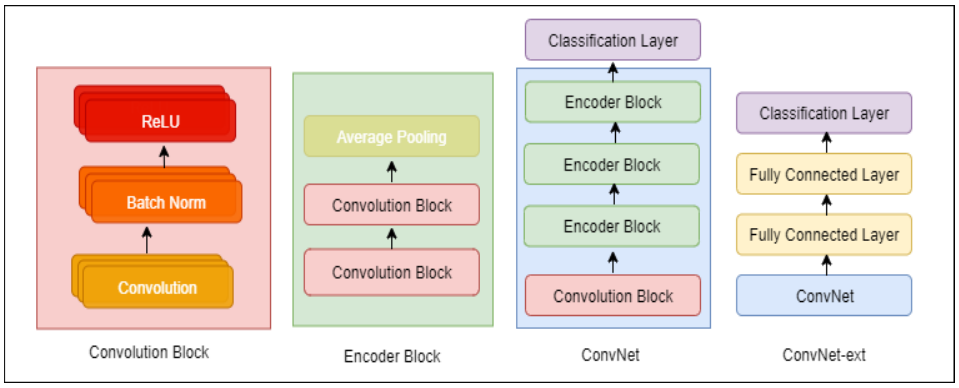


Figure 3: Model Architecture

6.2 Training Details

We use Adam optimizer [18] for the training the neural network. We also use a ReduceLROnPlateau scheduler to reduce the learning rate based on the validation set accuracy.

6.3 Evaluation

For the pretext task, we choose accuracy on the validation set as the evaluation metric for early stopping the network. For the downstream task, in addition to accuracy, we also check the F1 Score and the Precision-Recall curve of the classifier as this is a imbalanced class problem.

6.4 Loss Functions

6.4.1 Pretext Task

For both our pretext tasks which are binary classification problems, we use the binary cross entropy loss.

6.4.2 Downstream Task

For each datapoint we predict 20 binary labels, each representing the presence of one instrument. So this is a 20-class problem modelled as 20 binary class problems. We assume the classes are independent of each other. Therefore, we choose binary cross entropy loss and average it across all the 20 classes to get a cumulative loss to train the convnet for the downstream task. Binary cross entropy loss can be written as

$$\text{Loss} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{C}\sum_{k=1}^{C}-(ylog(p)+(1-y)log(1-p))$$

Since the data is partially labeled and the convnet is trained using mini-batch Stochastic Gradient Descent, some of the classes may be entirely missing from the batch. Also, the labels present in the batch may not be balanced. So we weigh the loss from each class label in each mini-batch according to the number of observations of the respective class labels present in that mini-batch.

7 Experiments and Results

7.1 Experiments

In order to improve the performance of the models on the pretext and downstream tasks as well as to understand the reasons behind the performance of our models, we ran multiple experiments. We set a random seed of 7 in all the libraries we used to make our experiments reproducible. We also avoided using CUDA based approximations that could make our experiments stochastic. These steps were all the more necessary due to the nature of Deep Neural Networks. They have been known to have non-convex objective functions. Below are the experiments and the outcomes of those experiments:

1. Varying the amount data used for training pretext tasks
2. Varying audio clip length in pretext tasks
3. Varying Architectures Used
4. Varying initialization and training methods for the downstream task
5. Varying amount of labelled data used for training downstream

7.1.1 Varying the amount data used for training pretext tasks

The FMA dataset comes in three variants namely FMA-small, FMA-medium and FMA-large. We used FMA-small for faster prototyping. After establing the entire ML pipeline and getting basic insights, we moved to FMA-large. FMA-small and FMA-large mainly differ in two aspects. First is the number of examples present in them 8,000 vs 106,574. Second is the variety of the data present in them. FMA-small contains very limited genres (8) as compared to FMA-large(161) (8 in small vs 161 in large).

Training on FMA-large is more expensive than training on FMA-small in terms of computational cost. However, training on FMA-large gives a boost in performance of our models. This is expected because of the wide variety and massive amount of data present in FMA-large. For example, in case of the jigsaw task, we could only achieve 72% validation accuracy using FMA-small while we achieved upto 88% using FMA-large. While for the reversal task, we saw a minor improvement in the performance from 95% to 96% and faster convergence rate by moving from FMA-small to FMA-large data.

7.1.2 Varying audio clip length in pretext tasks

- **Time Reversal**

  We vary length of the input audio clip to see the effect it has on the performance of the model for Time Reversal task. We believe that reducing the length of audio clip makes the Time Reversal task difficult. Our analysis performed on the FMA small dataset resonates with our intuition and we see an improvement in validation accuracy with the increase in length of audio clip. Having this insight and the fact that audio clip lengths in OpenMic are already 10 second long, we decided to fix 10 second as the ideal clip length for this task. The detailed loss curve diagrams for this task are present in the Appendix.

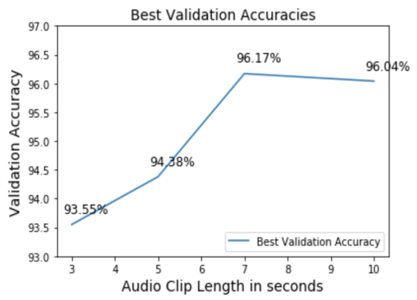


Figure 4: Effect of audio clip length on validation accuracy

- **Jigsaw**

  To decide on the audio clip length for the Jigsaw task, we had to consider the following question: 'If a human is given three small clips of audio, how many beats does each clip need to have so that the human is able to order the three clips correctly?' In most files in OpenMic have a tempo of 60 i.e. 1 beat per second. So, if the entire clip length is 10s, there are approximately 10 beats in the clip. If we further split that clip into three pieces, every sub-clip will have approximately 3 beats.

We experimented with two total clip lengths of 10s and 3s for the jigsaw task. So for every architecture we tested, we experimented with having three pieces of 3s each and three pieces of 1s each. Having three pieces of 3s each was always better in terms of accuracy. This is also inline with our intuition about human performance. If we are given three clips with just one beat each, it is hard for us to arrange them in order. However, we can do considerably better if we are given 3 clips with 3 beats each. So we decided that the total clip length should be 10 seconds for the jigsaw task as well. The detailed loss curve diagrams for this task are present in the Appendix.

7.1.3 Varying initialization and training methods for the downstream task

- **Initialization Method**

  As explained in the **Model Architecture** section, the downstream model has the same architecture as the pretext except for the final classification layer. We initialize the downstream model with weights from the trained pretext models. We initialize the final classification layer of the downstream model randomly. We refer to this downstream model as the self-supervised downstream model.

  In order to check the effectiveness of the self-supervised downstream model, we compare it with a randomly initialized model following the same architecture.

- **Training Strategies**

  We also experimented with freezing the weights of the model before training. In this method, we would train only the final classification layer. We can compare this method of training with training the whole network.

7.1.4 Varying amount of labelled data used for training downstream

Since SSL is most useful when there is a low amount of labeled data, we want to check the effect of the varying the amount of labeled data used to train the downstream self-supervised model. We use stratified sampling when sub-setting the labelled data to make sure we get a varied dataset that is representative of the actual class distribution in the OpenMic dataset. We experiment with the number of labeled examples per class in the range of $10, 50, 250, 500$ and the full training set. Figure 6 shows the performance of the model with the varying data size. We see that the model's performance increases with increase in data.

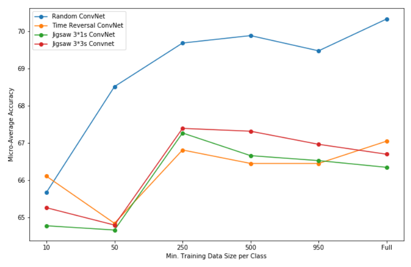
7.2 Results

Our pretext task results are presented in Table 1. The results are on our FMA validation set. For the Jigsaw task, we can see that ConvNet-ext architecture using 3*3 clip lengths achieved the best accuracy. For the Time Reversal task, ConvNet architecture was the only architecture we tried. 10 second audio clip achieves the best accuracy. The reason for not trying other architectures for Time Reversal was established in Section 6. We did not want to over-parameterize the model which already achieved 96% accuracy.

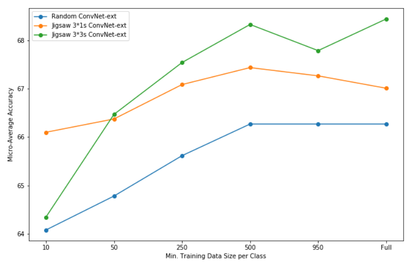
Pretext Task	Architecture	Pretext Audio Clip Length (in secs)	Validation Accuracy(%)
Jigsaw	ConvNet	$3 * 1$	50.23
Jigsaw	ConvNet	$3 * 3$	51.27
Jigsaw	ConvNet-ext	$3 * 1$	63.63
Jigsaw	ConvNet-ext	$3 * 3$	**88.69**
Jigsaw	ResNet18	$3 * 1$	81.26
Jigsaw	ResNet18	$3 * 3$	82.15
Time Reversal	ConvNet	5	94.54
Time Reversal	ConvNet	10	**96.33**

Table 1: Pretext Task Results

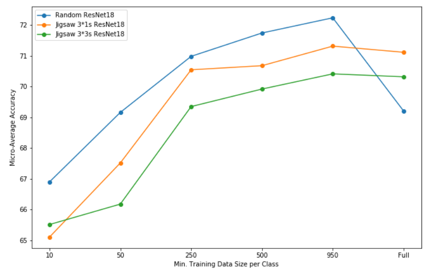
Our downstream task results are presented in Figure 6 and Table 2. The results pertain to models pretrained on FMA. The results are on test set of OpenMic. We can see that for each architecture we chose, randomly initialized neural networks either outperformed or performed similar to the self-supervised neural networks on the downstream task. There is no clear choice between the pretext tasks of Time Reversal and Jigsaw when comparing performance on the downstream. Models trained on both pretext tasks perform similarly. However, we see that models trained on 3*1 second jigsaw pieces perform worse than the neural networks trained on 3*3 seconds jigsaw pieces.



(a) Accuracy vs Size of training data for ConvNet



(b) Accuracy vs Size of training data for ConvNet-ext

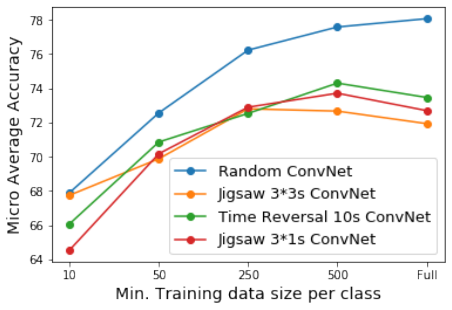


(c) Accuracy vs Size of training data for Resnet18

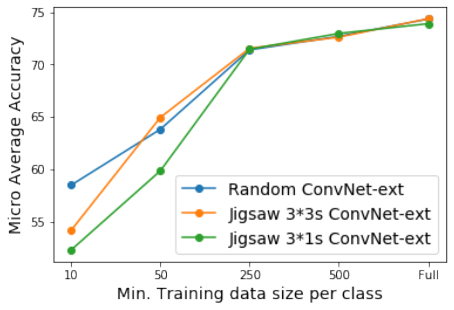
Figure 5: Accuracy vs Size of training data with frozen weights for different convnet architectures with different initializations

Initialization Method	**Architecture**	**Pretext Audio Clip Length (in secs)**	**Micro-Average Test Accuracy (in %)**	
			Frozen Feature Extractor	**Fine-Tuned Feature Extractor**
Random	ConvNet	-	70.33	**78.06**
Time Reversal	ConvNet	10	67.05	73.46
Jigsaw	ConvNet	$3 * 1$	66.35	72.79
Jigsaw	ConvNet	$3 * 3$	66.70	71.93
Random	ConvNet-ext	-	66.27	74.34
Jigsaw	ConvNet-ext	$3 * 1$	67.01	73.89
Jigsaw	ConvNet-ext	$3 * 3$	68.45	74.37
Random	ResNet18	-	69.21	74.29
Jigsaw	ResNet18	$3 * 1$	**71.12**	-
Jigsaw	ResNet18	$3 * 3$	70.32	75.40

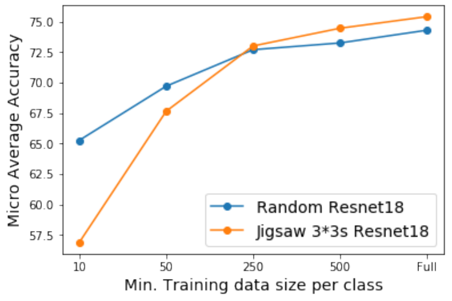
Table 2: Downstream Task Results



(a) Accuracy vs Size of training data for ConvNet



(b) Accuracy vs Size of training data for ConvNet-ext



(c) Accuracy vs Size of training data for Resnet18

Figure 6: Accuracy vs Size of training data with fine-tuning different convnet architectures with different initializations

7.3 Analysis of Results

7.3.1 Pretext Task Analysis

After training our models on the pretext task, we verified the predictions from the model to understand the model better. This gave us further insight into the reasons for the mistakes it made. This also served as a validation that the model's correct predictions were as expected. When analyzing the mistakes, we found some interesting results that we have discussed below.

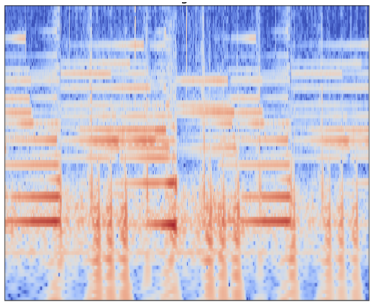
For the **Time Reversal** task, we saw that the incorrect predictions where the model was most confident that the music was played in reverse, the real musical piece was intentionally reversed by the artist. We show a sample spectrogram where the artist intentionally reverses the musical piece in Figure 7a. We also checked where the model was most confused i.e., the prediction of the model was near 0.5. We found that some instruments in a musical piece were being correctly played while some other were reversed. Also some of the music files were drone music Figure 7b where one note is sustained for a long time. As the sound lasts for a long time, there is no difference when it is reversed. This validated that our model was making sensible predictions.

For the **Jigsaw** task, we did the same analysis to see where the model made incorrect predictions and was most confident. We also saw where the model was confused. We found that the musical pieces where model was confused on whether the piece was jigsawed were confusing even to a human listener. Refer Figure 8.

7.3.2 Downstream Task Analysis

To understand the performance of the models on the downstream task, we check the Precision-Recall curve obtained from the best performing model on each of the 20 classes . In case of fine-tuning, the best model was the randomly initialized model trained on the full training data. Precision-Recall curves of some of the classifiers from the model are shared below.

We see that the model learns a classifier as good as a random classifier for Clarinet (figure 9a) while it learns a very good classifier for Drums (figure 9b). This is also visible from the F1 score obtained for the 2 classes. F1 score for Drums is 0.92 and for Clarinets is 0.01. The full Precision-Recall results are shared in table 3. In general, we see that as the ratio of positive to negative label decreases, F1 score of the classifier increases. This may be because we use Binary Cross Entropy loss which


(a) Spectrogram of music where artist intentionally reverses it

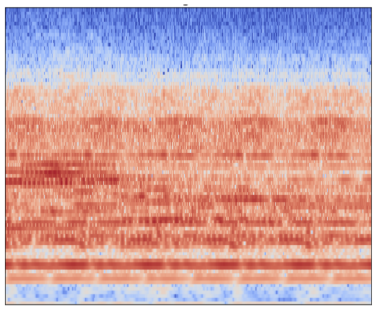

(b) Spectrogram of drone music

Figure 7: Example of incorrect predictions by the Time Reversal trained Model

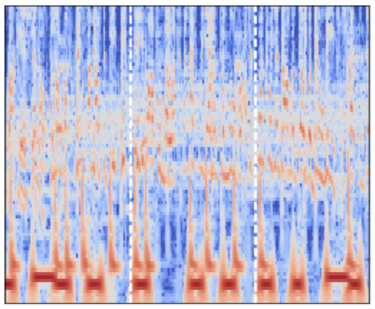
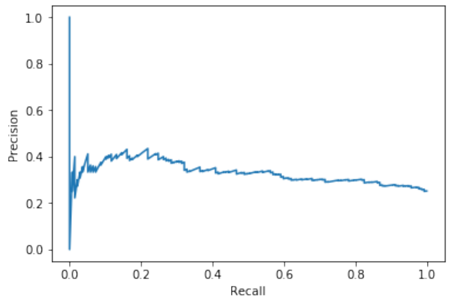
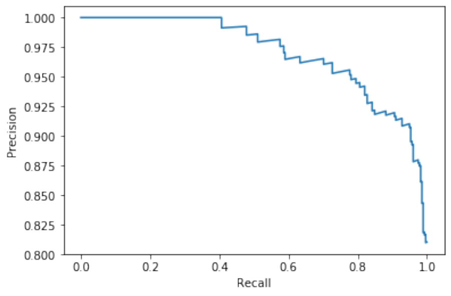


Figure 8: Example of incorrect prediction by the Jigsaw trained Model

weighs the positive and negative class equally, therefore the classifier gets more loss and gradients from the majority class thus focusing on predicting the majority classes more correctly.



(a) Classifier for Clarinets



(b) Classifier for Drums

Figure 9: Precision-Recall curves for 2 instrument classes

7.4 Limitations

Our analysis outlined in Section 7.3, gave us a good idea of the limitations of our approach.

From the pretext task analysis and class-specific downstream results, we could observe that our pretext tasks helped some instrument classes but were ineffective for the others. For example, the classification performance on instruments like cymbals and drums were considerably higher than performance on instruments like flutes and clarinets. On further thinking about the possible reasons behind this observation, we came to the conclusion that Time Reversal/Jigsaw might not be a suitable task for instruments like clarinets or flutes. A clarinet or a flute has long sustained notes and therefore does not have a lot of variations when reversed or jigsawed. The reversed spectrogram for a clarinet looks very similar to the original spectrogram. However, for other instruments like drums, the original

Instrument	Precision	Recall	F1 Score	# of Positive labels	# of Negative labels
Accordion	0.50	0.07	0.12	115	423
Banjo	0.52	0.39	0.45	140	338
Bass	0.79	0.51	0.62	134	329
Cello	0.72	0.70	0.71	226	259
Clarinet	0.25	0.01	0.01	137	503
Cymbals	0.91	0.95	0.93	297	139
Drums	0.88	0.96	0.92	278	387
Guitar	0.92	0.93	0.93	286	150
Mallet Percussion	0.66	0.59	0.62	211	267
Mandolin	0.54	0.39	0.45	193	434
Organ	0.62	0.35	0.44	121	310
Piano	0.93	0.97	0.95	285	130
Saxophone	0.70	0.76	0.73	305	324
Synthesizer	0.92	0.95	0.93	268	112
Trombone	0.67	0.38	0.49	228	492
Trumpet	0.69	0.56	0.61	318	467
Ukulele	0.69	0.48	0.56	182	408
Violin	0.81	0.89	0.85	394	237
Voice	0.86	0.92	0.89	224	150

Table 3: Precision, Recall and F1 Scores for the best fine-tuned model on test set

and reversed spectrograms are remarkably different. This might explain the difference in performance on different instruments.

This also motivates trying out other pretext approaches such as pretext invariant learning and contrastive predictive coding which we have elaborated on in Section 9.

Another limitation arises from the fact that the OpenMic dataset was sampled from FMA dataset. The sampling was done in a way to match the genre distribution of both datasets. The sampling mechanism was also engineered to get similar number of positively annotated data points for each class. [15]. Though the genre distribution of both datasets have been matched, the class distributions have not. This makes the class distribution of OpenMic different from that of FMA. We believe that this difference might cause a decrease in the effectiveness of self-supervising on the FMA dataset.

Our approach generalizes to other downstream tasks which use human generated music. Since our approach is to train on a huge repository of human generated music to learn representations of such music files, these representations will be useful in solving other related downstream tasks like genre identification, instrument tagging, pitch estimation. They can even be used in innovative tasks such as sequencing scrambled music into the correct order, generating artificial music, etc. However, care has to be taken to ensure the data distribution of the downstream task is similar to the data distribution of the FMA dataset.

In comparison, our approach will not be applicable to music from a different domain such as naturally occurring music. For example, BirdVOX is a dataset containing bird sounds. Bird sounds by nature are shorter and have a different spectra than human generated music. So self-supervision using FMA dataset will be ineffective on datasets like BirdVOX.

8 Conclusion

From our experiments, we can observe that our self-supervised pretraining improves the performance of two architectures, namely ConvNet-ext and ResNet, after crossing 250 labelled examples per class for supervised training. Refer Figures 6b and 6c. Our pretraining increase the validation accuracy of ConvNet-ext and ResNet by 2% over random initialization. From 6c we can say that with self-supervised pretraining, we are able to beat the performance of random initialization trained on the 'Full' labelled examples per class (11k labelled examples) using '500' labelled examples per class (7k labelled examples). This is a reduction of 40% annotated data which is an useful contribution of our project.

However, when amount of labelled data is lower than 250 examples per class, our pretraining cannot match the performance of random initialization. Therefore, our approach does not decrease the need for labelled data in this low labelled data region. This prompts further research investigating different pretext tasks and alternative datasets to reduce necessity of labelled data when using low labelled data. We have listed down a few approaches to carry forward the research in Section 9.

An interesting contribution of our research is that our pretext model is good at reasoning about sequence in a music clip. This is highlighted in Section 7.3. Our pretext Time Reversal model learns to distinguish between audio clips played in forward and reverse. Our pretext Jigsaw model learn to distinguish between naturally played music and shuffled music. Currently, the majorly used features in the domain of music information retrieval (MIR) such as MFCC (mel-frequency cepstral coefficient) cannot distinguish between a normal audio clip and reversed audio clip. Our two models serve as interesting feature extractors which can accomplish this. Therefore, they are useful in tasks such as sequence prediction, music remixing and restoration in MIR.

9 Future Work

Due to the novel, exploratory nature of our project, there are a lot of directions to extend this project. A potent direction of research can be exploring new pretext tasks.

- Pretext tasks using the temporal order in audio similar to video [19]
- Pretext invariant representation learning [20] is a new approach in learning representations that are invariant to the pretext tasks
- Recently, Contrastive Predictive Coding [21] has proved to be a successful pretext task in Image Recognition. It beat all the SoTa results on the ImageNet challenge.

An alternate direction would be test our models and strategies on a bioacoutsics dataset like BirdVox [22]. BirdVox is a dataset containing different bird sounds and the patterns in these kind of natural audio files are contrasting to a human generated dataset like FMA. So training self-supervised on naturally occurring data and testing it is definitely a worthwhile effort.

A radically different approach is to work with the music data in the time-amplitude domain instead of the frequency domain. In the time-amplitude domain, we can use the waveforms directly. A suitable pretext task would be inpainting [23], where a portion of the data is masked and the network is trained to predict the missing data.

Few techniques that can be tried to deal with issues concerning class imbalance are using weighted binary cross entropy loss or focal loss [24].

References

[1] Zisserman A. et al. *Self-Supervised Learning*. 1999. URL: `https://project.inria.fr/paiss/files/2018/07/zisserman-self-supervised.pdf` (visited on 12/07/2019).

[2] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *CoRR* abs/1803.07728 (2018). arXiv: `1803.07728`. URL: `http://arxiv.org/abs/1803.07728`.

[3] Donglai Wei et al. "Learning and Using the Arrow of Time". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 8052–8060.

[4] Mehdi Noroozi and Paolo Favaro. "Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles". In: *ECCV* (2016).

[5] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805 (2018). arXiv: `1810.04805`. URL: `http://arxiv.org/abs/1810.04805`.

[6] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: `1706.03762`. URL: `http://arxiv.org/abs/1706.03762`.

[7] H. Zhao C. Gan A. Rouditchenko et al. "The sound of pixels". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 570–586.

[8] A. Owens and A. A. Efros. "Audio-visual scene analysis with self-supervised multisensory features". In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 631–648.

[9] A. Owens J. Wu et al. “Ambient sound provides supervision for visual learning”. In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2016), pp. 801–816.

[10] B. Korbar D. Tran and L. Torresani. “Cooperative learning of audio and video models from self-supervised synchronization”. In: *Advances in Neural Information Processing Systems* (2018), pp. 7763–7774.

[11] R. Arandjelovi and A. Zisserman. “Look, listen and learn”. In: *Proceedings of the IEEE International Conference on Computer Vision)* (2017).

[12] R. Arandjelovi and A. Zisserman. “Objects that sound”. In: *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 435–451.

[13] M. Cartwright J. Cramer et al. “Tricycle: Audio Representation Learning from Sensor Network Data using Self-Supervision”. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics* (2019).

[14] *FREE MUSIC ARCHIVE*. URL: `https://freemusicarchive.org/`.

[15] Eric Humphrey, Simon Durand, and Brian McFee. “OpenMIC-2018: An Open Data-set for Multiple Instrument Recognition.” In: *ISMIR*. 2018, pp. 438–444.

[16] Lonce Wyse. “Audio Spectrogram Representations for Processing with Convolutional Neural Networks”. In: *CoRR* abs/1706.09559 (2017). arXiv: `1706.09559`. URL: `http://arxiv.org/abs/1706.09559`.

[17] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: `1512.03385`. URL: `http://arxiv.org/abs/1512.03385`.

[18] Diederik P Kingma and Jimmy Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014).

[19] Basura Fernando et al. “Self-Supervised Video Representation Learning With Odd-One-Out Networks”. In: *CoRR* abs/1611.06646 (2016). arXiv: `1611.06646`. URL: `http://arxiv.org/abs/1611.06646`.

[20] Ishan Misra and Laurens van der Maaten. “Self-Supervised Learning of Pretext-Invariant Representations”. In: *arXiv preprint arXiv:1912.01991* (2019).

[21] Olivier J. Hénaff et al. “Data-Efficient Image Recognition with Contrastive Predictive Coding”. In: *CoRR* abs/1905.09272 (2019). arXiv: `1905.09272`. URL: `http://arxiv.org/abs/1905.09272`.

[22] Vincent Lostanlen et al. “Birdvox-full-night: A dataset and benchmark for avian flight call detection”. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2018, pp. 266–270.

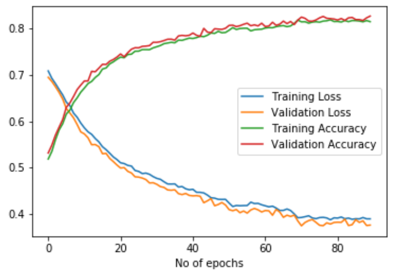
[23] Deepak Pathak et al. “Context Encoders: Feature Learning by Inpainting”. In: *CoRR* abs/1604.07379 (2016). arXiv: `1604.07379`. URL: `http://arxiv.org/abs/1604.07379`.

[24] Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *CoRR* abs/1708.02002 (2017). arXiv: `1708.02002`. URL: `http://arxiv.org/abs/1708.02002`.

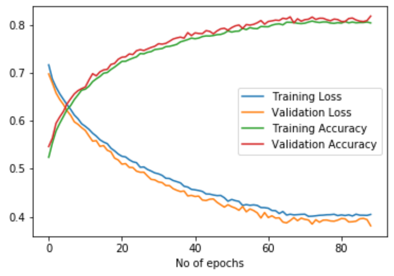
Appendix

9.1 Performance on Pretext Tasks

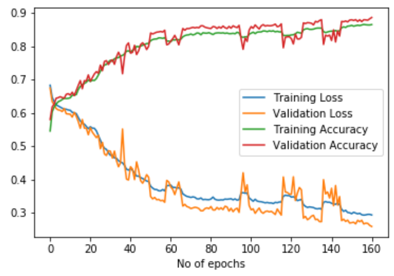
Here we present the detailed loss curves of the Jigsaw pretext training for different architectures. The performance on the varying clip lengths 7.1.2 has also been illustrated.



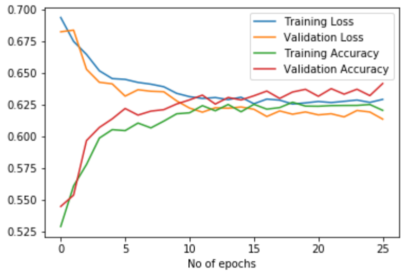
(a) ResNet architecture for 3*3s clips



(b) ResNet architecture for 3*1s clips



(c) ConvNet architecture for 3*3s clips



(d) ConvNet architecture for 3*1s clips

Figure 10: Performance of different architectures and varying clip lengths on the Jigsaw pretext task

Here we present the detailed loss curves of the Time Reversal pretext training for ConvNet architecture. The performance on the varying clip lengths 7.1.2 has also been illustrated.

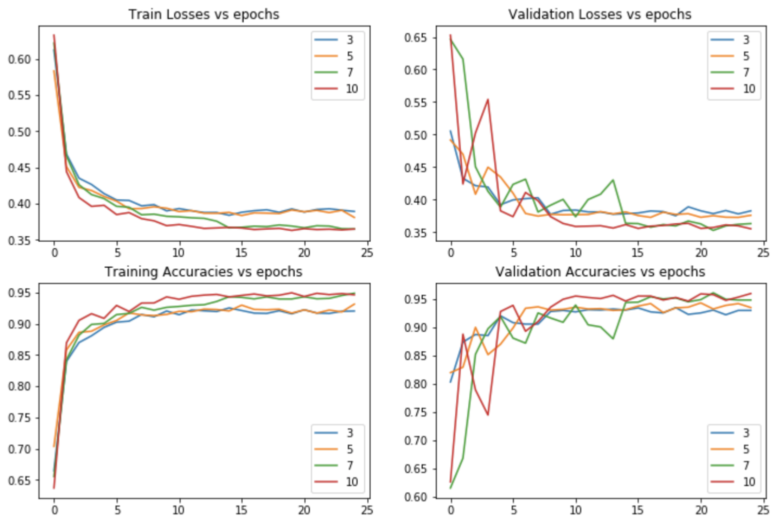


Figure 11: Performance of ConvNet architecture and varying clip lengths on the Time Reversal pretext task

9.2 Performance on Downstream Task

Here we present the accuracy, loss and F1-score curves of the various models with different architecture and initialization on validation set as observed during training.

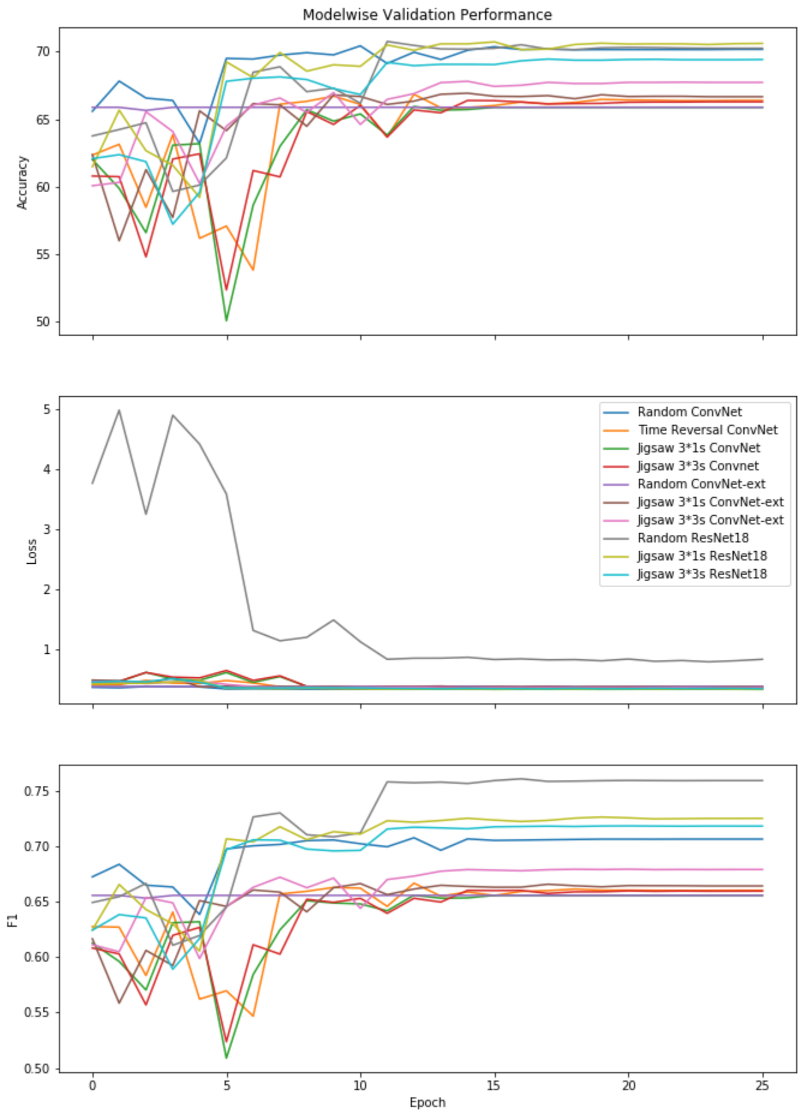


Figure 12: Performance of various models on validation set during training

Here we present the accuracy, loss and F1-score values of the the best model for each configuration with different architecture and initialization on validation and test set.

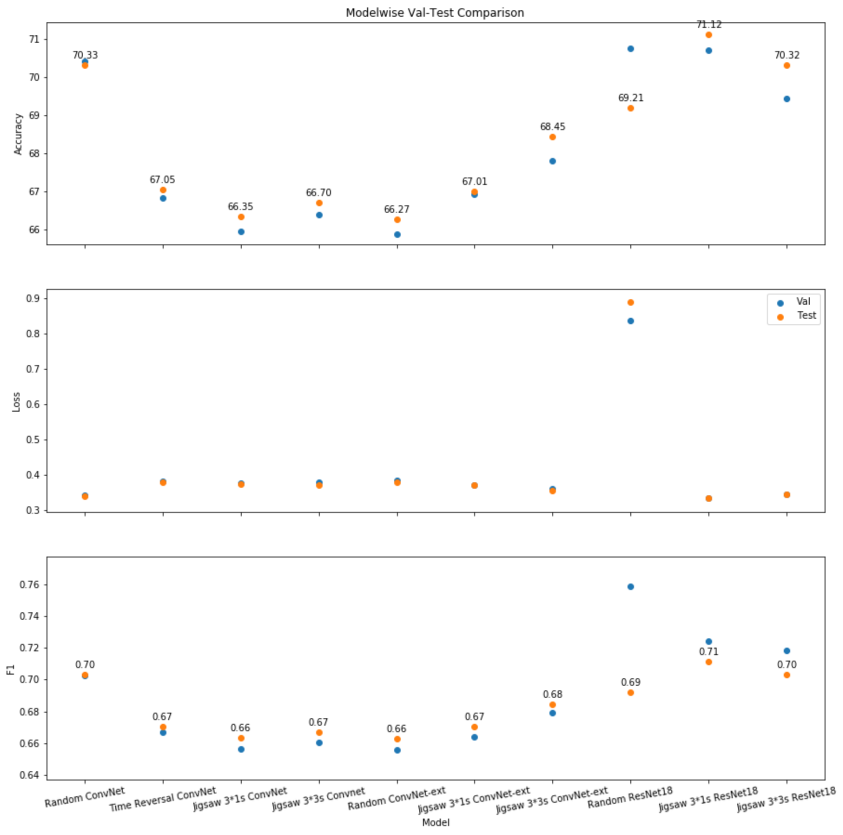


Figure 13: Performance of best model for each configuration on validation and test set