Temporal FiLM: Capturing Long-Range Sequence Dependencies with Feature-Wise Modulation

Sawyer Birnbaum\textsuperscript{1,2}, Volodymyr Kuleshov\textsuperscript{1,2}, S. Zayd Enam\textsuperscript{1}, Pang Wei Koh\textsuperscript{1}, and Stefano Ermon\textsuperscript{1}

\textsuperscript{1}Stanford University, Stanford, CA
\textsuperscript{2}Afesh Technologies, San Francisco, CA
\{sawyerb, kuleshov, zayd, pangwei, ermon\}@cs.stanford.edu

Abstract

Learning representations that accurately capture long-range dependencies in sequential inputs — including text, audio, and genomic data — is a key problem in deep learning. Feed-forward convolutional models capture only feature interactions within finite receptive fields while recurrent architectures can be slow and difficult to train due to vanishing gradients. Here, we propose Temporal Feature-Wise Linear Modulation (TFiLM) — a novel architectural component inspired by adaptive batch normalization and its extensions — that uses a recurrent neural network to alter the activations of a convolutional model. This approach expands the receptive field of convolutional sequence models with minimal computational overhead. Empirically, we find that TFiLM significantly improves the learning speed and accuracy of feed-forward neural networks on a range of generative and discriminative learning tasks, including text classification and audio super-resolution.

1 Introduction

In many application domains of deep learning — including speech [28, 20], genomics [2], and natural language [51] — data takes the form of long, high-dimensional sequences. The prevalence and importance of sequential inputs has motivated a range of deep architectures specifically designed for this data [21, 32, 52].

One of the challenges in processing sequential data is accurately capturing long-range input dependencies — interactions between symbols that are far apart in the sequence. For example, in speech recognition, data occurring at the beginning of a recording may influence the conversion of words enunciated much later.

Sequential inputs in deep learning are often processed using recurrent neural networks (RNNs) [15, 21]. However, training RNNs is often difficult, mainly due to the vanishing gradient problem [3]. Feed-forward convolutional approaches are highly effective at processing both images [34] and sequential data [32, 54, 17] and are easier to train. However, convolutional models only account for feature interactions within finite receptive fields and are not ideally suited to capture long-term dependencies.

In this paper, we introduce Temporal Feature-Wise Linear Modulation (TFiLM), a neural network component that captures long-range input dependencies in sequential inputs by combining elements of convolutional and recurrent approaches. Our component modulates the activations of a convolutional model based on long-range information captured by a recurrent neural network. More specifically, TFiLM parametrizes the rescaling parameters of a

\footnote{These authors contributed equally to this work.}
batch normalization layer as in earlier work on image stylization [12, 23] and visual question answering [45]. (Table 1 outlines recent work applying feature-wise linear modulation.)

We demonstrate real-world applications of TFiLM in both discriminative and generative tasks involving sequential data. We define and study a family of signal processing problems called time series super-resolution, which consists of reconstructing a high-resolution signal from low-resolution measurements. For example, in audio super-resolution, we reconstruct high-quality audio from a low-quality input containing a fraction (15-50%) of the original time-domain samples. Likewise, in genomic super-resolution, we recover high-quality measurements from experimental assays using a limited number of low-quality measurements.

We observe that TFiLM significantly improves the performance of deep neural networks on a wide range of discriminative classification tasks as well as on complex high-dimensional time series super-resolution problems. Interestingly, our model is domain-agnostic, yet outperforms more specialized approaches that use domain-specific features.

Contributions. This work introduces a new architectural component for deep neural networks that combines elements of convolutional and recurrent approaches to better account for long-range dependencies in sequence data. We demonstrate the component’s effectiveness on the discriminative task of text classification and on the generative task of time series super-resolution, which we define. Our architecture outperforms strong baselines in multiple domains and could, inter alia, improve speech compression, reduce the cost of functional genomics experiments, and improve the accuracy of text classification systems.

2 Background

Batch Normalization and Feature-Wise Linear Modulation. Batch normalization (batch norm; [24]) is a widely used technique for stabilizing the training of deep neural networks. In this setting, batch norm takes as input a tensor of activations $F \in \mathbb{R}^{T \times C}$ from a 1D convolution layer — where $T$ and $C$ are, respectively, the 1D spatial dimension and the number of channels — and performs two operations: rescaling $F$ and applying an affine transformation. Formally, this produces tensors $\hat{F}, F' \in \mathbb{R}^{T \times C}$ whose $(t,c)$-th elements are the following:

$$\hat{F}_{t,c} = \frac{F_{t,c} - \mu_c}{\sigma_c + \epsilon}$$

$$F'_{t,c} = \gamma_c \hat{F}_{t,c} + \beta_c$$

(1)

Here, $\mu_c, \sigma_c$ are estimates of the mean and standard deviation for the $c$-th channel, and $\gamma, \beta \in \mathbb{R}^C$ are trainable parameters that define an affine transformation.

Feature-Wise Linear Modulation (FiLM) [12] extends this idea by allowing $\gamma, \beta$ to be functions $\gamma, \beta : Z \rightarrow \mathbb{R}^C$ of an auxiliary input $z \in Z$. For example, in feed-forward image style transfer [12], $z$ is an image defining a new style; by using different $\gamma(z), \beta(z)$ for each $z$, the same feed-forward network (using the same weights) can apply different styles to a target image. [11] provides a summary of applications of FiLM layers.
Table 1: Recent work applying feature-wise linear modulation.

<table>
<thead>
<tr>
<th>Paper</th>
<th>Problem Area</th>
<th>Base Modality</th>
<th>Conditioning Modality</th>
<th>Conditioning Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dhingra et al. [9]</td>
<td>QA</td>
<td>Text (document)</td>
<td>Text (query)</td>
<td>CNN</td>
</tr>
<tr>
<td>Perez et al. [46] and de Vries et al. [7]</td>
<td>Visual QA</td>
<td>Images</td>
<td>Text</td>
<td>RNN/MLP</td>
</tr>
<tr>
<td>Dumoulin et al. [13]</td>
<td>Style Transfer</td>
<td>Images</td>
<td>Images</td>
<td>CNN</td>
</tr>
<tr>
<td>Kim et al. [31]</td>
<td>Speech</td>
<td>Audio</td>
<td>Self (Sequence)</td>
<td>Feedforward</td>
</tr>
<tr>
<td>Hu et al. [22]</td>
<td>Image Classification</td>
<td>Images</td>
<td>Self</td>
<td>Feedforward</td>
</tr>
<tr>
<td>This Paper</td>
<td>Sequence Analysis</td>
<td>Audio, Text, DNA</td>
<td>Self (Sequence)</td>
<td>RNN</td>
</tr>
</tbody>
</table>

Algorithm 1 Temporal Feature-Wise Linear Modulation.

**Input:** Tensor of 1D convolutional activations $F \in \mathbb{R}^{T \times C}$ where $T$, $C$ are, respectively, the temporal dimension and the number of channels, and a block length $B$. **Output:** Adaptively normalized tensor of activations $F' \in \mathbb{R}^{T \times C}$.

1. Reshape $F$ into a block tensor $F_{blk} \in \mathbb{R}^{B \times T/B \times C}$, defined as $F_{b,t,c} = F_{b \times B+t,c}$.
2. Obtain a representation $F_{pool}^{b,c} \in \mathbb{R}^{B \times C}$ of the block tensor by pooling together the channels within each block: $F_{b,c}^{pool} = \text{Pool}(F_{b,c}^{blk})$.
3. Compute sequence of normalizers $\gamma_b, \beta_b \in \mathbb{R}^{C}$ for $b = 1, 2, ..., B$ using an RNN applied to pooled blocks: $(\gamma_b, \beta_b), h_b = \text{RNN}(F_{b,c}^{pool}; h_{b-1})$ starting with $h_0 = \vec{0}$.
4. Compute normalized block tensor $F_{norm}^{b,c} \in \mathbb{R}^{B \times T/B \times C}$ as $F_{b,c}^{norm} = \gamma_b \cdot F_{b,c}^{pool} + \beta_b$.
5. Reshape $F_{norm}$ into output $F' \in \mathbb{R}^{T \times C}$ as $F'_{t,c} = F_{norm}^{\lfloor t/B \rfloor, t \mod B,c}$.

**Recurrent and Convolutional Sequence Models.** Sequential data is often modeled using RNNs [21, 39] in a sequence-to-sequence (seq2seq) framework [51]. RNNs are effective on language processing tasks over medium-sized sequences; however, on longer time series RNNs may become difficult to train and computationally impractical [54]. An alternative approach to modeling sequences is to use one-dimensional (1D) convolutions. While convolutional networks are faster and easier to train than RNNs, convolutions have a limited receptive field, and a subsequence of the output depends on only a finite subsequence of the input. This paper introduces a new layer that addresses these limitations.

### 3 Temporal Feature-Wise Linear Modulation

In this section, we describe a new neural network component called Temporal Feature-Wise Linear Modulation (TFiLM) that effectively captures long-range input dependencies in sequential inputs by combining elements of convolutional and recurrent approaches. At a high level, TFiLM modulates the activations of a convolutional model using long-range information captured by a recurrent neural network.

Specifically, let $F \in \mathbb{R}^{T \times C}$ be a tensor of activations from a 1D convolutional layer (at one datapoint, i.e. $N = 1$ and we drop the batch size dimension $N$ for simplicity); a TFiLM layer takes as input $F$ and applies the following series of transformations. First, $F$ is split along the time axis into blocks of length $B$ to produce $F_{blk} \in \mathbb{R}^{B \times T/B \times C}$. Intuitively, blocks correspond to regions along the spatial dimension in which the activations are closely correlated; for example, when processing audio, blocks could be chosen to correspond to audio samples that define the same phoneme. Next, we compute for each block $b$ an affine transformation parameters $\gamma_b, \beta_b \in \mathbb{R}^{C}$ using an RNN:

$$F_{b,c}^{pool} = \text{Pool}(F_{b,c}^{blk}) \in \mathbb{R}^{B \times C}, \quad (\gamma_b, \beta_b), h_b = \text{RNN}(F_{b,c}^{pool}; h_{b-1}) \text{ for } b = 1, 2, ..., B$$
starting with $h_0 = \vec{0}$, where $h_b$ denotes the hidden state, $F_{\text{pool}} \in \mathbb{R}^{B \times C}$ is a tensor obtained by pooling along the second dimension of $F_{\text{blk}}$, and the notation $F_{\text{blk}}^{b,c}$ refers to a slice of $F_{\text{blk}}$ along the second dimension. In all our experiments, we use an LSTM and max pooling.

Finally, activations in each block $b$ are normalized by $\gamma_b, \beta_b$ to produce a tensor $F_{\text{norm}}$ defined as

$$F_{\text{norm}}^{b,t,c} = \gamma_b^{c} \cdot F_{\text{block}}^{b,t,c} + \beta_b^{c}.$$ 

Notice that each $\gamma_b, \beta_b$ is a function of both the current and all the past blocks. Hence, activations can be modulated using long-range signals captured by the RNN. In the audio example, the super resolution of a phoneme could depend on previous phonemes beyond the receptive field of the convolution; the RNN enables us to use this long-range information.

Although TFiLM relies on an RNN, it remains computationally tractable because each RNN is small (the dimensionality of its output is only $O(C)$) and because the RNN is invoked only a small number of times. Consider again the speech example, where blocks are chosen to match phonemes: a 5 second recording contains $\approx 500.1$ second phonemes, yielding only about 50 RNN invocations for 80,000 audio samples at 16KHz. Despite being small, this RNN also carries useful long-range information, as our experiments demonstrate.

4 Time Series Super-Resolution

In order to demonstrate the real-world applications of TFiLM, we define and study a new generative modeling task called time series super-resolution, which consists of reconstructing a high-resolution signal $y = (y_1, y_2, ..., y_T)$ from low-resolution measurements $x = (x_1, x_2, ..., x_T)$; $x, y$ denote the source and target time series, respectively. For example, $y$ may be a high-quality speech signal while $x$ is a noisy phone recording.

This task is closely inspired by image super resolution [10, 36], which involves reconstructing a high-resolution image from a low-resolution version. As in image super-resolution, we assume that low- and high-resolution time series $x, y$ have a natural alignment, which can arise, for example, from applying a low-pass filter to $y$ to obtain $x$. Below, we provide two examples of time series super-resolution problems.

Audio Super-Resolution. Audio super-resolution (also known as bandwidth extension; [14]) involves predicting a high-quality audio signal from a fraction (15-50%) of its time-domain samples. Note that this is equivalent to predicting the signal’s high frequencies from its low frequencies. Formally, given a low-resolution signal $x = (x_1 / R_1, ..., x_{R_1T} / R_1)$ sampled at a rate $R_1 / T$ (e.g., low-quality telephone call), our goal is to reconstruct a high-resolution version $y = (y_1 / R_2, ..., y_{R_2T} / R_2)$ of $x$ that has a sampling rate $R_2 > R_1$. We use $r = R_2 / R_1$ to denote the upsampling ratio of the two signals. Thus, we expect that $y_{rt} / R_2 \approx x_{t} / R_1$ for $t = 1, 2, ..., TR_1$.

Super-Resolution of Genomics Experiments. Many genomics experiments can be seen as taking a real-valued measurement at every position of the genome; experimental results can therefore be represented by a time series. Measurements are generally obtained using a large set of probes (e.g., sequencing reads) that each randomly examine a different position in the genome; the genomic time series is an aggregate of the measurements taken by these probes. In this setting, super-resolution corresponds to reconstructing high-quality experimental measurements taken using a large set of probes from noisy measurements taken using a small set of probes. This process can significantly
reduce the cost of scientific experiments. This paper focuses on a particular genomic experiment called chromatin immunoprecipitation sequencing (ChIP-seq) [48].

4.1 A Deep Neural Network Architecture for Time Series-Super Resolution

The TFiLM layer is a key part of our deep neural network architecture shown in Figure 2. Other notable features of the architecture include the following: (1) a sequence of downsampling blocks that halve the spatial dimension and double the feature size and of upsampling blocks that do the reverse; (2) max pooling to reduce the size of LSTM inputs; (3) skip connections between corresponding downsampling and upsampling blocks; and (4) subpixel shuffling [49] to increase the time dimension during upscaling and avoid artifacts [42]. For more details, see the Appendix.

We train the model on a dataset $D = \{x_i, y_i\}_{i=1}^n$ of source/target time series pairs. As in image super-resolution, we take the $x_i, y_i$ to be small patches sampled from the full time series. We train the model using a mean squared error objective.

5 Experiments

In this section, we show that TFiLM layers improve the performance of convolutional models on both discriminative and generative tasks. We analyze TFiLM on sentiment analysis, a text classification task, as well as on several generative time series super-resolution problems.

5.1 Text Classification

Datasets. We use the Yelp-2 and Yelp-5 datasets [1], which are standard datasets for sentiment analysis. In the Yelp-2 dataset, reviews are classified as positive or negative based on the number of stars given by the reviewer. Reviews with 1 or 2 stars are classified as negative, and reviews with 4 or 5 stars are classified as positive. In the Yelp-5 dataset, reviews are labeled with their star rating (i.e., 1-5). The Yelp-2 dataset includes about 600,000 reviews and the Yelp-5 dataset includes 700,000 reviews. For both tasks, there are an equal number of examples with each possible label. With zero-padding and truncation, we set the length of Yelp reviews to 256 tokens.

Methods. We tokenize the reviews using Keras’s built-in tokenizer and use 100-dimensional GLoVe word embeddings [44] to encode the tokens.

We compare our method against a variety of recent work on the Yelp-2 and Yelp-5 datasets, and to a CNN model based on the architecture of Johnson and Zhang’s Deep Pyramid Convolutional Neural Network [26]. This “SmallCNN” model copies the DPCNN architecture, but reduces the number of convolutional layers to 3 and does not include region embeddings. Our full model inserts TFiLM layers after the convolutional layers in this architecture. We normalize the number of parameters between the basic SmallCNN model and the full model with TFiLM layers. To do so, we adjust the number of filters in the convolutional layers.

We train for 20 epochs using the ADAM optimizer with a learning rate of $10^{-3}$ and a batch size of 128.

Evaluation. Table 2 presents the results of our experiments. On both datasets, TFiLM layers improve the performance of the SmallCNN architecture; the resulting TFiLM model performs at or near the level of state-of-the-art methods from the literature. The models that outperform the TFiLM architecture use several times as many parameters (VDCNN and DenseCNN) or run unsupervised pre-training on external data (DPCNN, BERT, and XLNet). Additionally, the TFiLM model trains several times faster than the SmallCNN model or a comparably-sized LSTM. (See the Appendix for details.) Overall, these results indicate that TFiLM layers can provide performance and efficiency benefits on discriminative sequence classification problems.

5.2 Audio Super-Resolution

Datasets. We use the VCTK dataset [55] — which contains 44 hours of data from 108 speakers — and a Piano dataset — 10 hours of Beethoven sonatas [41]. We generate low-resolution audio signal from the 16 KHz originals by applying an order 8 Chebyshev type I low-pass filter before
We instantiate our model with a DNN (Deep Neural Network) based on the technique of Li et al., (in the high-resolution space) using the ADAM optimizer with a learning rate of $3 \times 10^{-4}$. To ensure source/target series are of the same length, the source input is pre-processed with cubic up-scaling. We adjust the TFiLM block length so that $T/B$ (the number of blocks) is always 32. We use a pooling stride and spatial extent of 8. To increase the receptive field of our convolutional layers, we use dilated convolutions with a dilation factor of 2.

Including TFiLM layers significantly increases the number of parameters per layer compared with the DNN baseline and the basic CNN architecture. Accordingly, we adjust the number of filters per layer to normalize the parameter counts between the models.

We instantiate our model with $K = 4$ blocks and train it for 50 epochs on patches of length 8192 (in the high-resolution space) using the ADAM optimizer with a learning rate of $3 \times 10^{-4}$. To ensure source/target series are of the same length, the source input is pre-processed with cubic up-scaling. We adjust the TFiLM block length $B$ so that $T/B$ (the number of blocks) is always 32. We use a pooling stride and spatial extent of 8. To increase the receptive field of our convolutional layers, we use dilated convolutions with a dilation factor of 2.

Including TFiLM layers significantly increases the number of parameters per layer compared with the DNN baseline and the basic CNN architecture. Accordingly, we adjust the number of filters per layer to normalize the parameter counts between the models.

**Methods.** We compare our method relative to three baselines: a cubic B-spline — which corresponds to the bicubic up-sampling baseline used in image super-resolution; a dense neural network (DNN) based on the technique of Li et. al., 2015 [37]; and a version of our CNN architecture without TFiLM layers.

We instantiate our model with $K = 4$ blocks and train it for 50 epochs on patches of length 8192 (in the high-resolution space) using the ADAM optimizer with a learning rate of $3 \times 10^{-4}$. To ensure source/target series are of the same length, the source input is pre-processed with cubic up-scaling. We adjust the TFiLM block length $B$ so that $T/B$ (the number of blocks) is always 32. We use a pooling stride and spatial extent of 8. To increase the receptive field of our convolutional layers, we use dilated convolutions with a dilation factor of 2.

Including TFiLM layers significantly increases the number of parameters per layer compared with the DNN baseline and the basic CNN architecture. Accordingly, we adjust the number of filters per layer to normalize the parameter counts between the models.

**Metrics.** Given a reference signal $y$ and an approximation $x$, the signal to noise ratio (SNR) is defined as $\text{SNR}(x, y) = 10 \log \frac{\|y\|^2}{\|x - y\|^2}$. The SNR is a standard metric used in the signal processing literature. The log-spectral distance (LSD) [19] measures the reconstruction quality of individual frequencies as follows $\text{LSD}(x, y) = \frac{1}{T} \sum_{t=1}^{T} \sqrt{\frac{1}{K} \sum_{k=1}^{K} (X(t, k) - \hat{X}(t, k))^2}$, where $X$ and $\hat{X}$ are the log-spectral power magnitudes of $y$ and $x$, respectively. These are defined as $X = \log |S|^2$, where $S$ is the short-time Fourier transform (STFT) of the signal. We use $t$ and $k$ index frames and frequencies, respectively; we used frames of length 8092.

**Evaluation.** The results of our experiments are summarized in Table 3. According to our SNR metric, our basic convolutional architecture shows an average improvement of 0.3 dB over the DNN and Spline baselines, with the strongest improvements on the SINGLE-SPEAKER task. Based on the LSD metric, the convolutional architecture also shows an average improvement of 0.5 dB over the DNN baseline and 1.6 dB over the Spline baseline. The convolutional architecture appears to use our modeling capacity more efficiently than a dense neural network; we expect such architectures will soon be more widely used in audio generation tasks.

Including the TFiLM layers improves performance by an additional 1.0 dB on average in terms of SNR and 0.2 dB on average in terms of LSD. The TFiLM layers prove particularly beneficial on the MULTI-SPEAKER task, perhaps because this is the most complex task and therefore the one which benefits most from additional long-term temporal connections in the model.

Finally, to confirm our results, we ran a study in which human raters assessed the quality of the interpolated audio samples. Our method ranked the best among the upscaling techniques; details are in the appendix.

---

2 We also experimented with a sequence-to-sequence architecture. This model performed very poorly, achieving SNR of about 0 dB for all upscaling ratios. As discussed above, sequence-to-sequence models generally struggle to solve problems involving extremely long time-series signals, as is the case here.

---

### Table 2: Text classification on Yelp-2 and Yelp-5 datasets.

<table>
<thead>
<tr>
<th>Method</th>
<th>Yelp-2</th>
<th>Yelp-5</th>
<th>Param</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastText [27]</td>
<td>95.7%</td>
<td>63.9%</td>
<td>Linear</td>
</tr>
<tr>
<td>LSTM [57]</td>
<td>92.6%</td>
<td>59.6%</td>
<td>-</td>
</tr>
<tr>
<td>Self-Attention [38]</td>
<td>93.5%</td>
<td>63.4%</td>
<td>-</td>
</tr>
<tr>
<td>CNN [32]</td>
<td>93.5%</td>
<td>61.0%</td>
<td>-</td>
</tr>
<tr>
<td>CharCNN [60]</td>
<td>94.6%</td>
<td>62.0%</td>
<td>-</td>
</tr>
<tr>
<td>VDCNN [6]</td>
<td>95.4%</td>
<td>64.7%</td>
<td>&gt;5M</td>
</tr>
<tr>
<td>DenseCNN [53]</td>
<td>96.0%</td>
<td>64.5%</td>
<td>&gt;4M</td>
</tr>
<tr>
<td>DPCNN* [26]</td>
<td>97.36%</td>
<td>69.4%</td>
<td>&gt;3M</td>
</tr>
<tr>
<td>BERT* [50][8]</td>
<td>98.11%</td>
<td>70.68%</td>
<td>-</td>
</tr>
<tr>
<td>XLNet* [56]</td>
<td>98.45%</td>
<td>72.2%</td>
<td>-</td>
</tr>
<tr>
<td>SmallCNN (ours)</td>
<td>78.1%</td>
<td>61.5%</td>
<td>&lt;1.5M</td>
</tr>
<tr>
<td>DPCNN+TFilm (full)</td>
<td>95.6%</td>
<td>62.3%</td>
<td>&lt;1.5M</td>
</tr>
</tbody>
</table>

---

**We also experimented with a sequence-to-sequence architecture. This model performed very poorly, achieving SNR of about 0 dB for all upscaling ratios. As discussed above, sequence-to-sequence models generally struggle to solve problems involving extremely long time-series signals, as is the case here.**
Table 3: Accuracy evaluation of audio super resolution methods (in dB) on each of the three super-resolution tasks at upscaling ratios \( r = 2, 4, \) and 8. DNN and CNN are baselines from the literature. [KEE17] denotes the convolutional method of Kuleshov et al. (2017).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( r = 2 )</td>
<td>SNR 19.0</td>
<td>19.0</td>
<td>19.4</td>
<td>19.5</td>
<td>18.0</td>
<td>17.9</td>
<td>18.1</td>
<td>19.8</td>
<td>24.8</td>
<td>24.7</td>
</tr>
<tr>
<td></td>
<td>LSD 3.5</td>
<td>3.0</td>
<td>2.6</td>
<td>2.5</td>
<td>2.9</td>
<td>2.5</td>
<td>1.9</td>
<td>1.8</td>
<td>1.8</td>
<td>2.5</td>
</tr>
<tr>
<td>( r = 4 )</td>
<td>SNR 15.6</td>
<td>15.6</td>
<td>16.4</td>
<td>16.8</td>
<td>13.2</td>
<td>13.3</td>
<td>13.1</td>
<td>15.0</td>
<td>18.6</td>
<td>18.6</td>
</tr>
<tr>
<td></td>
<td>LSD 5.6</td>
<td>4.0</td>
<td>3.7</td>
<td>3.5</td>
<td>5.2</td>
<td>3.9</td>
<td>3.1</td>
<td>2.7</td>
<td>2.8</td>
<td>3.2</td>
</tr>
<tr>
<td>( r = 8 )</td>
<td>SNR 12.2</td>
<td>12.3</td>
<td>12.7</td>
<td>12.9</td>
<td>9.8</td>
<td>9.8</td>
<td>9.9</td>
<td>12.0</td>
<td>10.7</td>
<td>10.7</td>
</tr>
<tr>
<td></td>
<td>LSD 7.2</td>
<td>4.7</td>
<td>4.2</td>
<td>4.3</td>
<td>6.8</td>
<td>4.6</td>
<td>4.3</td>
<td>2.9</td>
<td>4.0</td>
<td>3.5</td>
</tr>
</tbody>
</table>

# Params. | N/A | 6.72e7 | 7.09e7 | 6.82e7 | N/A | 6.72e7 | 7.09e7 | 6.82e7 | N/A | 6.72e7 | 7.09e7 | 6.82e7 |

5.3 Chromatin Immunoprecipitation Sequencing

We use histone ChIP-seq data from lymphoblastoid cell lines derived from several individuals of diverse ancestry [30] on the following common histone marks: H3K4me1, H3K4me3, H3K27ac, H3K27me3, and H3K36me3. This dataset contains high-quality ChIP-seq data with a high sequencing depth; to obtain low-quality versions, we artificially subsample 1M reads for each histone mark (out of the full dataset of 100+M reads per mark). This mirrors the setup of Koh et al., 2016 [33], which introduced a simpler convolutional neural network architecture. The Koh results are the state of the art for this task; we use them as a baseline in this section.

Formally, given an input noisy ChIP-seq signal \( X \in \mathbb{R}^{k \times T} \), where \( k \) is the number of distinct histone marks, and \( T \) is the length of the genome, we aim to reconstruct a high-quality ChIP-seq signal \( Y \in \mathbb{R}^T \). We use the \( k \) low-quality signals as input and train a separate model for each high quality target mark. We use \( B = 2 \) and training windows of length 1000; all other hyper-parameters are as in the audio-super resolution task.

To evaluate our results, we measure Pearson correlation between our model output and the true, high-quality ChIP-seq signal; this is a standard comparison metric in the field (e.g., [16]). As shown in Table 4, our method significantly improves the quality of the input signal over the Koh results, and on all but one histone mark outperforms the specialized CNN baseline. Across all of the histone marks, the model output from an input of 1M sequencing reads is equivalent in quality to signal derived from 10M to 20M reads, constituting a significant efficiency gain.

Table 4: Pearson correlation of the model output and the high-quality ChIP-seq signal derived from an experiment with high sequencing depth. [K17] indicates results from Koh et al. (2017); linear method performance is estimated.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>H3K4me1</td>
<td>0.37</td>
<td>0.41</td>
<td>0.59</td>
<td>0.79</td>
</tr>
<tr>
<td>H3K4me3</td>
<td>0.63</td>
<td>0.67</td>
<td>0.72</td>
<td>0.66</td>
</tr>
<tr>
<td>H3K27ac</td>
<td>0.55</td>
<td>0.61</td>
<td>0.77</td>
<td>0.85</td>
</tr>
<tr>
<td>H3K27me3</td>
<td>0.14</td>
<td>0.18</td>
<td>0.30</td>
<td>0.65</td>
</tr>
</tbody>
</table>

5.4 Additional Analyses

Model Visualization. We examined the internals of the TFiLM layer by visualizing the adaptive normalizer parameters in the audio super-resolution and sentiment analysis experiments. On the former, we observed that the parameters tend to cluster by gender, suggesting that the layer learns useful features. Figures are in the Appendix.

Ablation Analysis. The ablation analysis in Figure 5 indicates that temporal adaptive normalization significantly improves model performance. In addition, we performed an ablation analysis for the skip connections and found that they also significantly improve reconstruction accuracy. Our results are in the Appendix.
Model Generalization. We examined the extent to which the model generalizes across datasets. On the audio task, we observed a loss in performance when evaluating the model that was trained on speech on piano music (and vice versa). This highlights the need for diverse training data. Details are in the Appendix.

Missing Value Imputation. We experimented with imputing missing values from a sequence of daily grocery retail sales using various zero-out rates. TFiLM layers consistently provided performance benefits. Our full methodology and results are in the Appendix.

6 Previous Work and Discussion

Feature-Wise Linear Modulation. Previous work has applied feature-wise linear modulation to tasks including question answering, style transfer, and speech recognition (see Table 1). Our approach is most similar to that of Kim et al. [32], which modulates layer normalization parameters using a feed-forward model conditioned on an input audio sequence. Conversely, our method adjusts the batch normalization parameters of a feed-forward CNN using an RNN conditioned on the entire sequence. This significantly improves the CNN’s performance.

Time Series Modeling. In the machine learning literature, time series signals have most often been modeled with auto-regressive models, of which variants of recurrent networks are a special case [18, 39, 41]. Our approach generalizes conditional modeling ideas used in computer vision for tasks such as image super-resolution [10, 36] or colorization [59].

Applications to Audio and Genomics. Existing learning-based approaches include Gaussian mixture models [5, 43, 47], linear predictive coding [4], and neural networks [37]. Other recent work on audio super-resolution includes Wang et al.’s WaveNet model [54] and Macartney and Weyde’s Wave-U-Net model [40]. Our work proposes a new architecture, which scales better with data size and outperforms recent methods. Moreover, existing techniques involve hand-crafted features [47]; our approach is fully domain-agnostic. Statistical modeling of genomic data has been explored in population and functional genomics [35, 16]; our approach has the potential to make scientific experiments significantly more affordable.

Computational Performance. Our model is computationally efficient and can be run in real time. Unlike sequence-to-sequence architectures, our model does not require the complete input sequence to begin generating an output sequence.

7 Conclusion

In summary, our work introduces a temporal adaptive normalization neural network layer that integrates convolutional and recurrent layers to efficiently incorporate long-term information when processing sequential data. We demonstrate the layer’s effectiveness on three diverse domains. Our results have applications in areas including text-to-speech generation and sentiment analysis and could reduce the cost of genomics experiments.

Acknowledgments

This research was supported by NSF (#1651565, #1522054, #1733686), ONR (N00014-19-1-2145), and AFOSR (FA9550-19-1-0024)
References


