ENSEMBLE OF AUTO-ENCODER BASED AND WAVENET LIKE SYSTEMS FOR UNSUPERVISED ANOMALY DETECTION

Technical Report

Paweł Daniluk, Marcin Gośdziewski, Sławomir Kapka, Michał Kośmider

Samsung R&D Institute Poland
Artificial Intelligence
Warsaw, Poland
{p.daniluk, m.gozdziewsk, s.kapka, m.kosmider}@samsung.com

ABSTRACT

In this paper we report an ensemble of models used to perform anomaly detections for DCASE Challenge 2020 Task 2. Our solution comprises three families of models: Heteroskedastic Variational Auto-encoders, ID Conditioned Auto-encoders and a WaveNet like network. Noisy recordings are preprocessed using a U-Net trained for noise removal on training samples augmented with noised obtained from the AudioSet. Models operate either on OpenL3 embeddings or log-mel power spectra. Heteroskedastic VAEs have a non-standard loss function which uses model's own error estimation to weigh typical MSE loss. Model architecture i.e. sizes of layers, dimension of latent space and size of an error estimating network are independently selected for each machine type. ID Conditioned AEs are an adaptation of the class conditioned auto-encoder approach designed for open set recognition. Assuming that non-anomalous samples constitute distinct IDs, we apply the class conditioned auto-encoder with machine IDs as labels. Our approach omits the classification subtask and reduces the learning process to a single run. We simplify the learning process further by fixing a target for non-matching labels. Anomalies are predicted either by poor reconstruction or attribution of samples to the wrong machine IDs. The third solution is based on a convolutional neural network. The architecture of the model is inspired by the WaveNet and uses causal convolutional layers with growing dilation rates. It works by predicting the next frame in the spectrogram of a given recording. Anomaly score is derived from the reconstruction error. We present results obtained by each kind of models separately, as well as, a result of an ensemble obtained by averaging anomaly scores computed by individual models.

Index Terms— DCASE 2020 Task 2, Unsupervised anomaly detection, Machine Condition Monitoring, Conditioned Auto-Encoder, Variational Auto-Encoders, Heteroskedastic loss, OpenL3 embeddings, WaveNet

1. INTRODUCTION

Unsupervised anomaly detection is a problem of detecting anomalous samples under the condition that only non-anomalous (normal) samples have been provided during training phase. In this paper we focus on unsupervised anomaly detection in the context of sounds for machine condition monitoring – namely, is detecting mechanical failure by listening.

In this paper we report our solution to the DCASE 2020 Challenge Task 2 [1, 2, 3, 4]. We present three different approaches on tackling unsupervised anomaly detection problem. We describe Heteroskedastic Variational Auto-Encoder, ID Conditioned Auto-Encoder and WaveNet like methods in sections 2, 3 and 4 respectively. Finally, in the section 5 we combine those three methods in the form of an ensemble.

2. HETEROSKEDASTIC VARIATIONAL AUTO-ENCODER

2.1. Model

A typical Variational Auto-Encoder [5] comprises two networks: Encoder (E) which maps an input feature vector X to a normal distribution over the latent space ($\mathcal{N}(\mu, \sigma)$) where $\mu, \sigma = E(X)$ are k-dimensional vectors and k is a dimension of the latent space; and Decoder (D) which maps a latent vector Z to a reconstructed point in the feature space ($\hat{X} = D(Z)$). Sampling operation is performed on distributions returned by the encoder in order to obtain actual points in latent space which are fed to the decoder. Loss function is composed of two terms: $D_{KL}(\mathcal{N}(\mu, \sigma) \| \mathcal{N}(0, 1)) + \mathrm{MSE}(X, \hat{X})$. This is a special case of variational inference where an intractable probability distribution on feature space $p(X)$ is approximated by maximizing:

$$\log p(X) \geq \mathrm{E}_{q(Z|X)} \log p(X|Z) + D_{KL}(q(Z|X) \| p(Z))$$

Distribution $q(Z|X)$ is normal with diagonal covariance matrix and parameters computed by the encoder, $p(Z)$ is $\mathcal{N}(0, 1)$, and $p(X|Z)$ is normal with mean given by the decoder ($\hat{X}$) and unit variance. The latter results with the $\mathrm{MSE}(X, \hat{X})$ term in the loss function.

In our solution we propose to improve upon vanilla VAE by changing the structure of $p(X|Z)$. We replace unit variance with an additional output of the decoder. Thus the decoder estimates not only reconstruction ($\hat{X}$) but also its precision – $\hat{X}, \hat{\sigma} = D(Z)$. Conditional probability $p(X|Z)$ takes the form:

$$p(X|Z) = p(X|D(Z)) = p(X|\hat{X}, \hat{\sigma}) = \frac{1}{\sqrt{2\pi\hat{\sigma}^2}} e^{-\frac{1}{2}\sum_{i=1}^{n} \frac{(x_i - \hat{x}_i)^2}{\hat{\sigma}_i^2}} \quad (1)$$

Which after omitting unnecessary constants leaves us with the following loss function:

$$\mathcal{L}(X, \mu, \sigma, \hat{X}, \hat{\sigma}) = D_{KL}(\mathcal{N}(\mu, \sigma) \| \mathcal{N}(0, 1)) + \\ + \beta(\mathrm{wMSE}(X, \hat{X}, \hat{\sigma}) - \sum_{i=1}^{n} \log \hat{\sigma}_i) \quad (2)$$

where β is an additional parameter which helps to achieve balance between loss components [6].

We have determined experimentally that wMSE, which is high for samples for which reconstruction is unexpectedly poor, is the best performing anomaly score, when training model on normal data only.

2.2. Data preparation

In the preprocessing we used a denoising network based on Deep Complex U-Net for noise removal from the original samples [7, 8]. The main premise of the network is to estimate a complex mask and apply it to the spectrogram.

The model is trained to extract clean sounds from a noisy sample by minimizing difference between a clean sample and a denoised one. In order to reduce background factory sounds and make the machine of interest more audible a separate model is trained for each machine type. In each case clean samples were obtained by extracting various mechanical sounds from the AudioSet [9]. Normal samples of other machine types were used as background and mixed with clean samples to obtain noisy samples. This approach is justified by the assumption that different machine types have completely different sounds, and thus a sample containing factory noise mixed with sounds of a particular machine resembles background noise when considering denoising a different one.

During training models are saved at some intervals corresponding to different values of SNR in order to obtain denoisers that can be used for different levels of denoising. As a result, there are 10 models for machine type. A single model contains 2.6M parameters.

2.3. Model architecture and ensembles

We have built several models with slightly varying architectures and latent space dimensions. In each model encoder and decoder have the same number of hidden layers. Encoder has two outputs for each latent space dimension. A softplus function is applied to σ to make it non-negative. There are two variants of a decoder with different methods of estimating $\hat{\sigma}$. It is either estimated by a doubled output layer (as in encoder) or by a separate network of the same architecture (i.e. all layers are doubled). We call the second approach *Big Precision* (*BP*). All layers except output have ELU activation and dropout with 0.1 rate.

We trained models separately for each machine type on all available samples (both development and additional datasets). OpenL3 embeddings or log-mel power spectra on 5 consecutive frames (as in the baseline model) were used as features. Anomaly score was calculated for each frame independently. We tried the following methods of score averaging over frames:

- *mean*
- *median*
- mean after computing medians in a 3 sec. window (*winmed*)
- mean after capping results within 3 standard deviations calculated on all frames for a given machine type (*meancap*)

Table 1: HVAE models

Type	Features	Den.	lev.	Hidden layers	Z	BP	β	Ep.	Averaging
TCar	OL3	3		[512, 512, 256]	16	+	1.0	400	winmed
TCar	OL3	3		[512, 512, 256]	16		2.0	500	winmed
TCar	OL3	1		[512, 512, 256]	16		4.0	200	winmed
TCar	OL3	6		[512, 512, 256]	16		2.0	400	winmed
TCar	OL3	6		[512, 512, 256]	16	+	2.0	400	winmed
TCar	LM	9		[640, 512, 256]	16	+	1.0	500	winmed
TCar	LM	7		[640, 512, 256]	16		1.0	500	winmed
TCar	LM	7		[640, 512, 256]	16	+	1.0	100	winmed
TCar	LM	5		[640, 512, 256]	16	+	1.0	200	winmed
TCar	LM	6		[640, 512, 256]	16	+	2.0	100	winmed
TConv.	OL3	-		[1024, 512]	16		2.0	500	winmed
TConv.	LM	-		[640, 512, 256]	20	+	4.0	200	winmed
fan	OL3	5		[512, 256, 256]	8	+	1.0	50	median
fan	OL3	9		[512, 256, 256]	8	+	2.0	50	median
fan	OL3	7		[512, 256, 256]	8	+	4.0	200	meanlim
fan	LM	-		[640, 256, 256]	20	+	4.0	100	meanlim
fan	LM	2		[640, 256, 256]	20		4.0	100	median
fan	LM	-		[640, 256, 256]	20		1.0	50	meanlim
pump	OL3	-		[512, 256, 256]	16		1.0	500	median
pump	OL3	-		[512, 256, 256]	16	+	1.0	300	meanlim
pump	OL3	-		[512, 256, 256]	8		2.0	100	median
slider	LM	-		[640, 512, 256]	20		2.0	400	mean
slider	LM	-		[640, 512, 256]	20	+	4.0	400	mean
slider	LM	1		[640, 512, 256]	20		2.0	200	mean
valve	OL3	9		[512, 256, 256]	16	+	4.0	500	meanlim
valve	OL3	9		[512, 256, 256]	16		4.0	500	meanlim
valve	OL3	9		[512, 256, 256]	10		1.0	500	meanlim
valve	LM	2		[640, 512, 256]	20	+	4.0	300	mean
valve	LM	3		[640, 512, 256]	20	+	2.0	300	mean
valve	LM	4		[640, 512, 256]	20	+	4.0	400	mean

Scores of the selected models were averaged to achieve the final anomaly prediction.

Table 1 lists all models used to build an ensemble submitted as `Daniluk_SRPOL_task2_1`.

3. ID CONDITIONED AUTO-ENCODER (IDCAE)

In this section we introduce the ID Conditioned Auto-Encoder (IDCAE), which is an adaptation of the Class Conditioned Auto-Encoder [10] designed for the open-set recognition problem [11].

3.1. Proposed Method

Given a machine type, we assume that we have a various IDs of the machine. In the nomenclature from [10], we treat machines with different IDs as distinct classes.

Our system constitutes of three main parts:

- encoder $E : \mathcal{X} \to \mathcal{Z}$ which maps *feature vector* X from input space $\mathcal{X}$ to the *code* $E(X)$ in the latent space $\mathcal{Z}$,

- decoder $D : \mathcal{Z} \to \mathcal{X}$ which takes the code Z from $\mathcal{Z}$ and outputs the vector $D(Z)$ of the same shape as feature vectors from $\mathcal{X}$,
- conditioning made of two functions $H_\gamma, H_\beta : \mathcal{Y} \to \mathcal{Z}$ which take the one-hot label l from $\mathcal{Y}$ and map it to the vectors $H_\gamma(l), H_\beta(l)$ of the same size as codes from $\mathcal{Z}$.

During feed-forward, the code Z is combined with $H_\gamma(l), H_\beta(l)$ to form $H(Z, l) = H_\gamma(l) \cdot Z + H_\beta(l)$. Thus, our whole system takes two inputs X, l from $\mathcal{X}$ and $\mathcal{Y}$ respectively and outputs $D(H(E(X), l))$.

Given an input X with some ID, we call label corresponding to this ID by the *match* and all other labels by *non-matches*. We wish that our system reconstructs X faithfully if and only if it is a normal sample conditioned by the matching label. Anomalies are predicted either by poor reconstruction or attribution of samples to the wrong IDs.

Given an input X, we set the label l to the match with probability α or to a randomly selected non-match with probability $1 - \alpha$, where α is predefined. Thus, for a batch $X_1, X_2, \dots X_n$ approximately α fraction of samples will be conditioned by matches and $1 - \alpha$ by non-matches. If l is the match, then the loss equals difference between the system's output and X, that is $\|D(H(E(X), l)) - X\|$. If l is a non-match, then the loss equals difference between the system's output and some predefined constant vector C with the same shape as X, that is $\|D(H(E(X), l)) - C\|$. In our setting $\|\cdot\|$ is either L_1 or the square of L_2 norm.

During the inference we always feed the network with matching labels. If a sample is non-anomalous, we expect the reconstruction to be faithful resulting in low reconstruction error. If the sample is anomalous, there may be two cases. If the sample is nothing like any sample during training, then auto-encoder wouldn't be able to reconstruct it resulting in high reconstruction error. However, if the sample reminiscent normal samples from the other IDs, then the auto-encoder will try to reconstruct the vector C resulting again in high error.

3.2. Model Architecture

In our model, we feed the network with fragments of normalised log-mel power spectrograms. Feature vector space $\mathcal{X}$ consists of vectors of the shape (F, M), where F is the frame size and M is the number of mels. Given an audio signal we first compute its Short Time Fourier Transform with 1024 window and 512 hop size, we transform it to power mel-scaled spectrogram with M mels, and we take its logarithm with base 10 and multiply it by 10. Finally, we standardize all spectrograms frequency-wise to zero mean and unit variance, and sample frames of size F as an input to our system.

As described in subsection 3.1, our model constitutes of the encoder E, the decoder D and the conditioning H_γ, H_β. In our case all these component are fully connected neural networks. Thus, we have to flatten feature vectors and reshaped the output to (F, M) for the sake of the dimension compatibility. The dense layers in E and D are followed by batch-norm and relu activation function, while the dense layers in H_γ, H_β are followed just by sigmoid activation functions. E has three hidden dense layers with $128, 64$ and 32 units followed by the latent dense layer with 16 units. D is made of four hidden dense layers each with 128 units. H_γ and H_β have both a single hidden dense layer with 16 units. We summarise the architecture in the Table 2.

Table 2: The architecture of IDCAE

Encoder (E)	**Decoder** (D)	**Conditioning** (H_γ, H_β)
Input (F, M)	Input 16	Input #classes
Flatten	DenseBlock 128	Dense 16
DenseBlock 128	DenseBlock 128	sigmoid
DenseBlock 64	DenseBlock 128	Dense 16
DenseBlock 32	DenseBlock 128	
DenseBlock 16	Dense $F \cdot M$	
	Reshape (F, M)	
DenseBlock n: Dense n, Batch-norm, relu		

Table 3: The scores of the IDCAE.

Machine	**Baseline**		**Dev**		**Dev+Add**	
Score	AUC	pAUC	AUC	pAUC	AUC	pAUC
Toy car	78.77	67.58	78.67	73.91	88.87	85.67
Toy conveyor	72.53	60.43	69.85	58.83	68.62	58.82
Fan	65.83	52.45	76.90	69.29	79.29	74.88
Pump	72.89	59.99	78.45	71.09	84.54	77.76
Slide rail	84.76	66.53	79.28	66.84	81.25	68.49
Valve	66.28	50.98	76.26	54.67	82.21	56.46
Average	73.34	59.66	76.57	65.77	80.80	70.35

We train our models using Adam optimizer with default parameters [12]. For each machine, we train our network for 100 epochs with exponential learning rate decay by multiplying the learning rate by 0.95 every 5 epochs. For every epoch we randomly sample 300 frames from each spectrogram.

3.3. Submissions

For our first submission `Kapka_SRPOL_task2_2` we unified the hyperparameters for all the machines. Namely, we set $\alpha = 0.75, C = 5$ with $F = 10$, $M = 128$ and trained our models using mean absolute error. We conducted two distinct experiments for this setup. In the first one, we trained our system just on the train split from the development dataset. In the second one, we train ours system on the train splits from both development and additional datasets. In fact, training on more IDs provides a better performance. We summarise the results in Table 3.

For our second submission, we made an ensemble for each machine using for training the train splits from both development and additonal datasets. We set $F = 10$ and done a grid search with $\alpha \in \{0.9, 0.75, 0.5\}, C \in \{0, 2.5, 5, 10\}, M \in \{128, 256\}$ trying mean square and mean absolute errors. We selected 4 models for each machine that maximize average of AUC and pAUC with $p = 0.1$ on the test split from the development dataset, and for each machine we done an ensemble by selecting 4 weights such that the weighted anomaly score maximize average of AUC and pAUC. We forward the output further to the ensemble `Daniluk_SRPOL_task2_4`, which is described in section 5.

4. KOSMIDER_SRPOL_TASK2_3 (FREAK)

This submission was inspired by WaveNet [13], but applied in the frequency domain instead of the time domain. In this approach, a neural network predicts the next frame in the spectrogram of a recording of interest. The difference between the prediction and

Table 4: The scores of the individual methods and the ensemble

Machine	**Baseline**		**HVAE**		**IDCAE**		**Freak**		**Ensemble**	
Score	AUC	pAUC	AUC	pAUC	AUC	pAUC	AUC	pAUC	AUC	pAUC
Toy car	78.77	67.58	93.39	85.46	91.25	87.36	96.73	89.30	98.30	93.55
Toy conveyor	72.53	60.43	83.46	68.98	72.23	60.23	87.22	72.59	89.02	73.89
Fan	65.83	52.45	80.94	66.55	81.82	76.98	93.51	85.10	94.12	88.23
Pump	72.89	59.99	85.26	74.35	88.17	80.36	95.87	89.53	97.31	92.56
Slide rail	84.76	66.53	95.64	90.74	86.49	74.65	97.36	94.60	97.85	94.54
Valve	66.28	50.98	91.54	77.10	84.59	62.41	97.94	91.58	98.35	92.11
Average	73.34	59.66	88.37	77.19	84.09	73.66	94.77	87.11	95.82	89.14

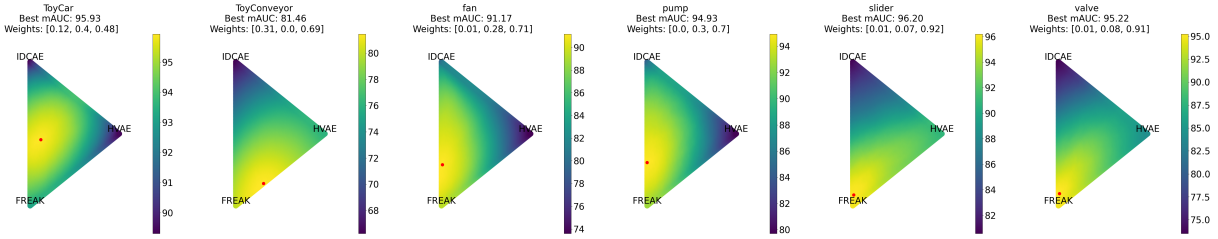

Figure 1: Red dots indicates the weights for which mAUC (the average of AUC and pAUC) is maximized.

Table 5: Architecture of the Freak model, variant with three layers.

layer	**channels**	**dilation**	**kernel**	**groups**
ResidualBlock	m*bins	1	3	4
ResidualBlock	m*bins	2	3	4
ResidualBlock	m*bins	4	3	4
CausalConv1D	bins	8	3	4

the actual frame is used to detect malfunctions. Architecture (Table 5) is based on one dimensional causal convolutions, and frequency bins are treated as channels. Channels are split into four groups/bands and processed separately, which is usually referred to as grouped convolution [14]. Residual blocks follow that of WaveNet and contain one causal convolution, followed by two parallel convolutions (gate and value) that are then multiplied together and a final convolution (see Figure 4 in WaveNet [13]). The skip connection is processed by a convolution without normalization or activation. In a residual block all convolutions other the the causal convolution have kernel of size one. GroupNorm [15] is used for normalization with a single group (which makes it equivalent to LayerNorm [16]). Normalization is applied before each convolutional layer. Padding is not added and therefore first few frames of a spectrogram are not predicted. Activation is applied just after each convolution. Last layer has no activation. Sigmoid activation is used for gates and ReLU is used everywhere else.

The final submission is an ensemble for which models have varying settings. Each device type has four to fourteen models. Most have three or four layers and hidden layers have five to six times as many channels as the input spectrogram. The spectrograms are created using STFT with window of 2048 samples and hop length of 512 samples, which is then transformed to mel spectrograms with either 64 or 128 frequency bins. Logarithm is applied to the resulting mel spectrogram (except for valve, where square root is applied or nothing at all). Subsequently each frequency is standardized independently using statistics from the training dataset. All models are trained for a specific device type, but using all available machines of that type together. Models are not given any information about the specific machine ID. Adam [12] optimizer is used for training with $\alpha = 0.001$, $\beta_1 = 0.85$ and $\beta_2 = 0.999$. Batch size is set to thirty two.

The loss is simply the mean squared error loss. However, scores for the predictions are computed differently. First method relies on percentiles and is used for valve and slider. Firstly for each frequency bin the 95$^{\text{th}}$ percentile of the squared error is computed (across time), then again 95$^{\text{th}}$ percentile of these (across frequencies). Second method relies on the negative log probability of the difference between the amplitude spectra of the prediction and the actual recording. This approach is less sensitive to symmetric noise. Amplitude spectrum is computed based on the spectrogram averaged over time. The probability is estimated using a multivariate normal distribution fitted during training using Welford's online algorithm [17]. The covariance matrix is either full (for ToyCar), restricted to diagonal (for ToyConveyor) or identity (for fan and pump).

5. ENSEMBLE

We generated our last submission by combining methods from sections 2, 3 and 4. For each machine, we standardize anomaly scores and using grid search we select 3 weights such that weighted anomaly score maximizes the average of AUC and pAUC on the test split from the development dataset. The actual weights are ilustrated in Figure 1. Our final submission `Daniluk_SRPOL_task2_4` is the weighted mean of these standardized anomaly scores. The results of individual models and the ensemble are summarised in Table 4.

6. REFERENCES

[1] Y. Koizumi, S. Saito, H. Uematsu, N. Harada, and K. Imoto, "ToyADMOS: A dataset of miniature-machine operating sounds for anomalous sound detection," in *Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)*, November 2019, pp. 308–312. [Online]. Available: https://ieeexplore.ieee.org/document/8937164

[2] H. Purohit, R. Tanabe, T. Ichige, T. Endo, Y. Nikaido, K. Suefusa, and Y. Kawaguchi, "MIMII Dataset: Sound dataset for malfunctioning industrial machine investigation and inspection," in *Proceedings of the Detection and Classification of Acoustic Scenes and Events 2019 Workshop (DCASE2019)*, November 2019, pp. 209–213. [Online]. Available: http://dcase.community/documents/workshop2019/proceedings/DCASE2019Workshop_Purohit_21.pdf

[3] Y. Koizumi, Y. Kawaguchi, K. Imoto, T. Nakamura, Y. Nikaido, R. Tanabe, H. Purohit, K. Suefusa, T. Endo, M. Yasuda, and N. Harada, "Description and discussion on DCASE2020 challenge task2: Unsupervised anomalous sound detection for machine condition monitoring," in *arXiv e-prints: 2006.05822*, June 2020, pp. 1–4. [Online]. Available: https://arxiv.org/abs/2006.05822

[4] http://dcase.community/challenge2020/task-unsupervised-detection-of-anomalous-sounds.

[5] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, p. 307–392, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000056

[6] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner, "Understanding disentangling in β-vae," 2018.

[7] H.-S. Choi, J.-H. Kim, J. Huh, A. Kim, J.-W. Ha, and K. Lee, "Phase-aware speech enhancement with deep complex u-net," 2019.

[8] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," 2017.

[9] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017.

[10] P. Oza and V. M. Patel, "C2ae: Class conditioned auto-encoder for open-set recognition," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] W. J. Scheirer, A. de Rezende Rocha, A. Sapkota, and T. E. Boult, "Toward open set recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 7, pp. 1757–1772, 2013.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[13] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499 [cs]*, Sept. 2016, arXiv: 1609.03499. [Online]. Available: http://arxiv.org/abs/1609.03499

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, May 2017. [Online]. Available: https://dl.acm.org/doi/10.1145/3065386

[15] Y. Wu and K. He, "Group Normalization," *arXiv:1803.08494 [cs]*, June 2018, arXiv: 1803.08494. [Online]. Available: http://arxiv.org/abs/1803.08494

[16] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer Normalization," *arXiv:1607.06450 [cs, stat]*, July 2016, arXiv: 1607.06450. [Online]. Available: http://arxiv.org/abs/1607.06450

[17] B. P. Welford, "Note on a Method for Calculating Corrected Sums of Squares and Products," *Technometrics*, vol. 4, no. 3, pp. 419–420, Aug. 1962.