ENVIRONMENTAL SOUND CLASSIFICATION WITH SEMI-SUPERVISED LEARNING

Rasmus Arpe Fogh Jensen (s134843), Thomas Pethick (s144448)

Technical University of Denmark

ABSTRACT

In this paper, we provide preliminary work on a semi-supervised learning approach for environmental sound classification. A convolutional autoencoder is used for pre-training the weights in the network. Two different methods for the invertions of the max-pooling layers are examined in the decoder; upsampling and unpooling. The semi-supervised approach is benchmarked against a supervised approach with similar architecture on the public available dataset ESC-50. The ESC-US dataset is used for unsupervised pre-training. The results show that the autoencoder learns useful features leading to semi-supervised learning yielding slightly better performance utilizing the unlabeled data. For the accompanying code we refer to `https://github.com/Rasmusafj/02456-deep-learning-project12/tree/paper`.

Index Terms— Environmental sound, classification, semi-supervised learning, convolutional neural networks, autoencoder

1. INTRODUCTION

The application of deep learning methods in the context of speech and audioprocessing has recently gained a lot of attention. The models often rely on a log-scaled mel-spectrogram representation of audio samples. Google has however shown that it is possible to obtain state of the art performance for some audioprocessing tasks using convolutions directly on the raw audio feed with their proposed Wavenet model [1].

Research in classification of environmental sounds (everyday audio events that do not consist of music or speech) has been conducted with results showing that deep learning methods are able to outperform previously suggested methods such as unsupervised feature learning with spherical k-means [2] for some datasets [3, 4]. The state-of-the-art models perform convolutions on the mel-spectrogram representation of the sound data with a fully connected layer at the top and also utilize feature engineering by performing various data-augmentations [4].

A common problem with deep learning for sound classification is the lack of labeled data. Google has recently introduced the Audioset [5] dataset in order to combat the problem and the dataset can be exploited in transfer-learning methods. When the domain of the sounds is substantially different from the Audioset, transfer learning might not necessarily yield the best results. Companies often have unlabeled data available for their particular domain of interest. An interesting field of research is therefore how we can utilize the unlabeled data for pre-training in order to achieve better classification results.

In this article, we explore the performance of a semi-supervised learning approach on an environmental sound classification task. We use convolutional autoencoders as a way of pre-training the weights in the network to obtain better discrimination. Two different structures for the decoding part of the autoencoder are explored; upsampling and unpooling. We benchmark our semi-supervised method against a convolutional architecture with dense layers on top. The results are obtained using public available datasets.

The article is organized as following. Section 2 describes our proposed approach to the problem. Section 3 describes the datasets used and the pre-proccessing required. Section 4 explains how the experiments were conducted and the hyper-parameter optimization. Section 5 shows and discusses the results of the approach. At last, we conclude in section 6 and consider further work.

2. METHOD

2.1. Semi-supervised learning

Semi-supervised learning is a class of supervised learning where training makes use of unlabeled data in addition to the labeled data. Useful features can be learned using the unlabeled data that can improve classification.

More precisely, training is done on n unlabeled examples $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}$ and m examples $\boldsymbol{x}^{(n+1)}, ..., \boldsymbol{x}^{(n+m)}$ with corresponding labels $\boldsymbol{y}^{(n+1)}, ..., \boldsymbol{y}^{(n+m)}$. The goal is to achieve a higher classification accuracy using both datasets than what would be possible by discarding the unlabeled data and doing supervised learning.

2.2. Autoencoder

An autoencoder is suitable for semi-supervised learning. An autoencoder consists of an encoder and a decoder where the

Thanks to Corti for supervising the project

encoder maps the input to a hidden representation and the decoder then reconstructs the input based on that hidden representation [6]. To be precise, a single layer autoencoder takes an input $\boldsymbol{x} \in \mathbb{R}^d$ encodes it to $\boldsymbol{z} \in \mathbb{R}^{d'}$ using a deterministic function $\boldsymbol{z} = \sigma(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b})$. The input is then reconstructed using a reverse mapping $\widetilde{\boldsymbol{x}} = \sigma(\boldsymbol{W'}\boldsymbol{z} + \boldsymbol{b'})$.

The weights are trained by minimizing the difference between $\boldsymbol{x}$ and $\widetilde{\boldsymbol{x}}$ called the reconstruction error [7],

$$E_{rec} = \frac{1}{2n}\sum_{i=1}^{n} ||\widetilde{\boldsymbol{x}}^{(i)} - \boldsymbol{x}^{(i)}||^2.$$

To avoid that the autoencoder simply learn the identity function some regularization is needed. These methods often involves introducing a regularizing term in the cost function that imposes sparsity. We will however enforce sparsity through the network architecture as suggested by [8] and covered in section 2.3.

Instead of using a single layer as encoder and decoder, we can extend the autoencoder to a deeper structure by stacking layers as in section 2.3.

The autoencoder will be used in the semi-supervised learning approach by providing weight initialization for part of the final classification network. First the autoencoder is trained on the unlabeled data. The encoder is then extended with a classifier and trained on the labeled data. This final step is called *fine-tuning* and adjusts all weights in the network. We will refer to the classifier initialized with weights from the autoencoder as being *pre-trained*.

2.3. Convolutional Autoencoder (CAE)

Instead of a fully connected autoencoder, the encoder can consist of convolutions as proposed by [8]. The main observation is that a max-pooling layer in the hierarchical network can be used to impose sparsity so the encoder does not learn a trivial identity function even if the hidden representation is of higher dimensionality than the input. By having the max-pooling layer, the need for regularizers like L1 and L2 are avoided.

The authors also experimented with adding noise to the input as a regularizing effect (but of course still evaluating the cost function on the original input). However, this did not seem to lead to more biological plausible filters compared to max-pooling, so we have avoided this additional complexity.

Originally the authors did the unsupervised pre-training of the stacked convolutional autoencoder in a greedy, layer-wise fashion. Subsequently, good results have been achieved on networks as deep as the ones we consider by training the whole network at once [9]. Consequently, we have decided to train all weights simultaneously.

2.4. Unpooling

The question of how to construct the decoder arises when the encoder is a combination of convolutions and max-pooling layers.

To create a function mapping from $\boldsymbol{z}$ to $\widetilde{\boldsymbol{x}}$ one approach is to reverse all layers. A transposed convolution is commonly used in the decoder as originally proposed in [10]. We refrain from using the term deconvolution as it is not the inverse of convolution. For a thorough explanation of the differences we refer to [11, 12].

It is however not obvious how max-pooling should be reversed. In the encoder, downsampling occured whereas the decoder has to upsample. Max-pooling is a many-to-one relationship and the location in the input of the picked output value is not necessarily known when doing upsampling. The naive approach uses the input by repeating each value to neighboring fields in the output which we will refer to as *upsampling*.

Zhao et al. [13] proposes a different solution that records the position used in the max-pooling layer. Using this position the corrosponding layer in the decoder then only copies the input value to the output on that location. All other output values are set to a value of 0. The authors argues that this approach, which we will refer to as *unpooling*, provides a regularization effect and thus leads to better generalization and reconstructions.

Since an implementation neither existed in Tensorflow, Theano or Keras at the time of writing we have made an implementation of an unpooling layer in Keras[1]. However, note that this layer is currently only supported by the Theano backend since it requires taking the gradient of the gradient which is not available in Tensorflow.

3. DATA

All of the results are obtained using the datasets ESC-50 and ESC-US [14].

ESC-50 is a collection of 2000 labeled environmental sounds manually extracted from public field recordings gathered from `Freesound.org`. The dataset includes sounds such as animal sounds, water sounds and urban sounds. Baseline classifiers using *k-nearest neighbour* (K-NN) and *random Forest* (RF) included in [14] performs with 32% and 45% accuracy, respectively. However, [3] obtained an accuracy of 62% using a convolutional neural network approach.

ESC-US is a collection of 250,000 unlabeled sounds also extracted from public field recordings of `Freesound.org`. They are however not manually annotated. ESC-US is suitable for unsupervised pre-training.

[1] The implementation is available at `https://github.com/Rasmusafj/02456-deep-learning-project12/blob/paper/layers.py`

All of the sounds from ESC-50 and ESC-US datasets are of 5 seconds length sampled at 44.1kHz.

3.1. Pre-processing

The sounds were normalized and processed into the frequency domain of log-scaled mel-spectrograms using the Librosa [15] python library. The mel-spectrograms were extracted using 28 mel-bands, window size of 12288 and hop-length of 6144. This gives in an input dimension of $\mathbf{x}^{(n)} \in \mathbb{R}^{28\times 36}$. The mel-spectrogram transformation involves a small mel-bandwidth and a large window size compared to results from [3, 4], hence some features important for classification might not be present. Furthermore, we restrict the pre-training to 50000 samples of ESC-US. The choice of these parameters is based on faster training and iteration since the time scope for the project was limited. However, the architecture and the pre-processing is easily scalable for future experimentation.

The Pescador [16] library is used to handle streams of data fed to the neural networks.

4. EXPERIMENTATION

We adopt the notation used in [13] to describe the considered architectures. The notation `(64)5c-2p-50fc` should be read as a 5x5 convolutional kernel with 64 filters, followed by a 2x2 max-pooling layer, and finally ending with a fully connected layer with 50 hidden units. When used to specify an autoencoder, only the encoder is described while the decoder is implied by that specification. Unpooling is assumed in the decoder when nothing else is explicitly stated.

4.1. Activation

The rectified linear unit (ReLU) is used for activation throughout the network except in the last layer of the decoder where a linear activation is needed for the mean squared error function and the last layer of the classifier, where a softmax function is used.

4.2. Early Stopping

In addition to the regularization effect of max-pooling we employ early stopping to avoid over-fitting. This is a method that stops the training loop when the validation loss no longer decreases. A patience of 5 is used to allow for fluctuations especially in the beginning before the loss starts converging.

4.3. Reconstructions

When training the CAE the goal is to achieve good reconstructions. Such reconstruction for test samples using both upsampling and unpooling are shown on figure 1. This provides an indication that unpooling might work better but it is not certain that a better representation is learned. For this reason both upsampling and unpooling is explored for pre-training the network.

4.4. Architecture

The architecture is inspired by the VGG-16 based autoencoder used for semantic segmentation which leverages unpooling [17]. However, it adopts the architecture used for MNIST in original paper on unpooling by Zhao et al. [8] since the input size is in the same order of magnitude. The small squared convolutional kernels used in this architecture are performant even for mel-spectograms [18, 19].

The base architecture used for the autoencoder was `(f)5c-2p-(f)3c-2p-(f)3c-2p` with batch size 32 using unpooling as seen in figure 3 with the corresponding classifier illustrated in figure 2. Exhaustive grid search was employed for batch sizes $16, 32, 64$ and number of filters $f \in \{16, 32, 64\}$. The biggest filter size 64 yielded better results in all cases. This is expected as long as we regularize properly since the model is more expressive. Bigger filters were not considered because it would drastically extend training time.

The remaining parameters were chosen heuristically due to the extensive training time required. The Adam optimizer[20] was used with an initial learning rate of 0.001 after observing that 0.01 did not converge after several epochs. We experimented with a smaller network of only 2 convolutional layers but this yielded a dramatic decrease in performance so we stopped pursuing that direction.

Upsampling was quickly discarded. A pre-trained CNN network using CAE with upsampling achieve a classification accuracy of 0.018 after 7 epochs of pre-training and 29 epochs of classification training. Without pre-training the same network achieved an accuracy of 0.41 in 25 epochs. Considering this is approximately the prediction accuracy of randomly guessing this seems suspiciously bad. It might be stuck in some local optimum where the learning rate is not big enough to adjust the weights sufficiently fast for it to learn over the 30 epochs. Thus we have not pursued it further after attempting different hyperparameters. However, it is still likely that some configuration of hyperparameters might yield a useful model based on upsampling.

Overfitting did seem to be an issue with training accuracy reaching 100% for the best performing models. Batch normalization was tested as regularization to avoid overfitting. However, this did not have positive results. A possible explanation for this will be given in section 5.

5. RESULTS

The final performance was slightly improved by using pre-trained weights as is seen in table 1. This aligns with the findings that unsupervised pre-training helps by supporting better generalization achieved through guiding the learning towards

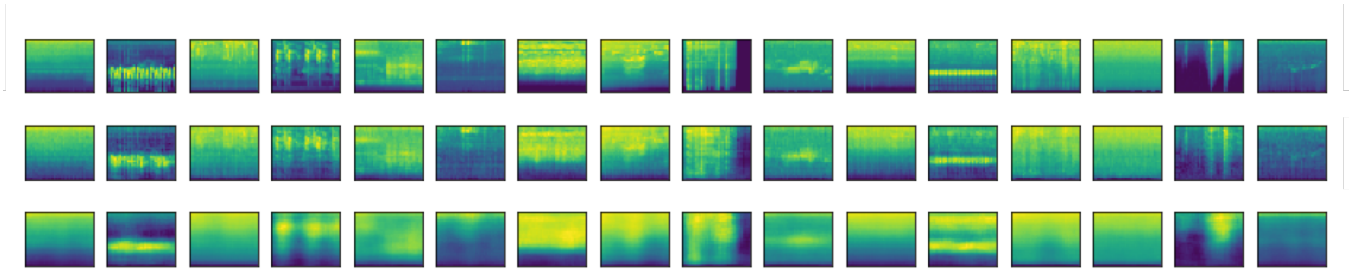

Fig. 1. Display of reconstruction on test set using `(64)5c-2p-(64)3c-2p-(64)3c-2p-50fc`. From top to bottom: 1) ground truth 2) reconstruction using unpooling 3) reconstruction using upsampling.

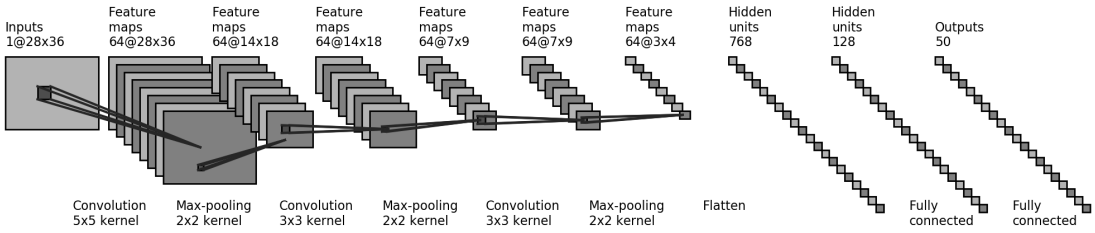

Fig. 2. Final CNN classifier Architecture.

basins of minima [21]. However, the increase in performance is only minimal for the best performing model and it is difficult to conclude that it is significant with no cross-validation performed.

Plotting the accuracies during training showed that for all combinations of experimented parameters, the training accuracy was converging with the validation accuracy always lower but steadily increasing. We noticed that most of the classifiers with pre-trained weights started with a validation accuracy around 0.1. This indicates that the weights are cleverly initialized, which is the purpose of the semi-supervised learning apprach.

Considering the small input size and the limited regularization, the performance for the pre-trained weights model is surprisingly good. It is not the case that one category is hugely over-represented and neither does the confusion matrix[2] show that a subset of the classes are responsible for the accuracy. The good accuracy with little regularization might indicate that the training data and testing data are too similar.

Architecture		Accuracy	
b	f	Supervised	Pre-training with CAE
16	64	0.414	0.502
32	64	**0.492**	**0.507**
64	64	0.474	-

Table 1. Accuracy results for architecture `(f)5c-2p-(f)3c-2p-(f)3c-2p` where `f` is the number of filters and `b` is the batch size.

We have shown that a model with pre-trained weights slightly outperforms a model without pre-training of weights. The performance is however still not comparable with what Piczak achieves [3]. To reach a similar level of accuracy, we expect it to be crucial to increase the input size. This will be discussed further in section 6.1.

[2]The confusion matrix for the classifier with pre-trained weights can be found in complimentary jupyter notebook.

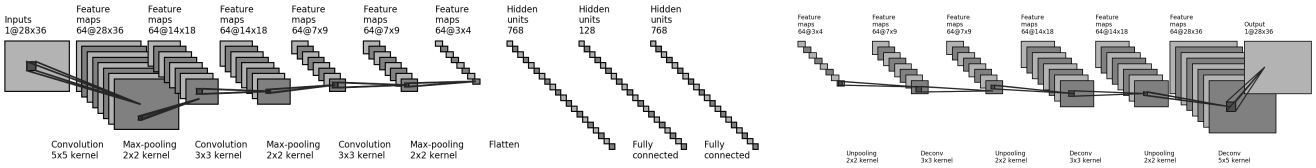

Fig. 3. Final CAE Architecture.

Network	Accuracy
Baseline Random Forest	44.0%
Piczak CNN	65.5%
Our CAE pre-trained CNN	50.7%

Table 2. Accuracy baseline and CNN results obtained by Piczak [3] compared with for pre-trained `(64)5c-2p-(64)3c-2p-(64)3c-2p-128fc-50fc`.

6. CONCLUSION

In this paper, we showed the architecture and results of a semi-supervised learning approach for environmental classification on the ESC-50 dataset using the ESC-US dataset for unsupervised pre-training. We tried two different convolutional autoencoders with different methods for inverting the max-pooling layer, namely upsampling and unpooling. The unpooling approach provided better reconstructions as seen in figure 1 but is not necessarily an indicator of a better initialization of weights. In section 5, the results showed a slightly better accuracy with pre-trained weights as seen in table 2 but cross-validation and increase of complexity in input size is needed to conclude a significant improvement. Finally, further regularization is needed for better generalization since overfitting was observed by inspecting the accuracy during training.

6.1. Further Work

We notice two immediate areas where further work is needed and desribe a few interesting derivations of the method explored.

As touched in section 5 more data exploration is necessary to verify the performance of the architecture. This includes doing k-fold cross-validation so we can be confident that the pre-trained model is significantly better.

Additionally, we expect the small input size to be the main course for the lack of performance. Usually a much higher band size is used as well as smaller window size leading to more frames as evident in [3, 4]. It is highly likely that the smaller spectograms will lack crucial features from the original waveform input necessary to distinguish between classes. However, the proposed architecture has the potential of scaling to bigger input sizes. Combined with training on all 250,000 unlabeled samples this would be the natural next step when considering how the accuracy could be improved.

So far, upsampling and switched based unpooling have been tested for the decoder. Promising results has been achieved in segmentation and super-resolution with different AE architectures such as U-Net [22] and efficient Sub-Pixel CNNs [23] respectively. We anticipate that an augmented version of these models for audio could speed up training time and potentially thus lead to better performance.

When scaling the input it is obvious to consider the pre-trained VGG-16 network build for large-scale audio classification [19]. It seems feasible that pre-training could improve such a network similarly to what have seen done with the VGGNet for images [24]. This however ignores the initial motivation that training data in the particular domain might not match the data used to train the audio VGG-16 model. The possibility is nonetheless interesting to pursue.

Acknowledgements

We would like to thank Lars Maale from Corti for providing supervision of the project.

7. REFERENCES

[1] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[2] Justin Salamon and Juan Pablo Bello, "Unsupervised feature learning for urban sound classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 171–175.

[3] Karol J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*, New York, NY, USA, 2015, MM '15, pp. 1015–1018, ACM.

[4] Justin Salamon and Juan Pablo Bello, "Deep convolutional neural networks and data augmentation

for environmental sound classification," *CoRR*, vol. abs/1608.04363, 2016.

[5] Jort F. Gemmeke, Daniel P. W. Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R. Channing Moore, Manoj Plakal, and Marvin Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *Proc. IEEE ICASSP 2017*, New Orleans, LA, 2017, Retrieved October 25, 2017 from `https://static.googleusercontent.com/media/research.google.com/da//pubs/archive/45857.pdf`.

[6] Wei Luo, Jun Li, Jian Yang, Wei Xu, and Jian Zhang, "Convolutional sparse autoencoders for image classification," *IEEE transactions on neural networks and learning systems*, 2017.

[7] Wu Haiyan, Yang Haomin, Li Xueming, and Ren Haijun, "Semi-supervised autoencoder: A joint approach of representation and classification," in *Computational Intelligence and Communication Networks (CICN), 2015 International Conference on*. IEEE, 2015, pp. 1424–1430.

[8] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," *Artificial Neural Networks and Machine Learning–ICANN 2011*, pp. 52–59, 2011.

[9] Maximilian Kohlbrenner, Russell Hofmann, Sabbir Ahmmed, and Youssef Kashef, "Pre-training cnns using convolutional autoencoders," .

[10] Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus, "Deconvolutional networks," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2528–2535.

[11] Wenzhe Shi, Jose Caballero, Lucas Theis, Ferenc Huszar, Andrew Aitken, Christian Ledig, and Zehan Wang, "Is the deconvolution layer the same as a convolutional layer?," *arXiv preprint arXiv:1609.07009*, 2016.

[12] Vincent Dumoulin and Francesco Visin, "A guide to convolution arithmetic for deep learning," *arXiv preprint arXiv:1603.07285*, 2016.

[13] J Zhao, M Mathieu, R Goroshin, and Y Lecun, "Stacked what-where auto-encoders. arxiv 2015," *arXiv preprint arXiv:1506.02351*.

[14] Karol J. Piczak, "Esc: Dataset for environmental sound classification," in *Proceedings of the 23rd ACM International Conference on Multimedia*, New York, NY, USA, 2015, MM '15, pp. 1015–1018, ACM.

[15] Brian McFee, Matt McVicar, Oriol Nieto, Stefan Balke, Carl Thome, Dawen Liang, Eric Battenberg, Josh Moore, Rachel Bittner, Ryuichi Yamamoto, Dan Ellis, Fabian-Robert Stoter, Douglas Repetto, Simon Waloschek, CJ Carr, Seth Kranzler, Keunwoo Choi, Petr Viktorin, Joao Felipe Santos, Adrian Holovaty, Waldir Pimenta, and Hojin Lee, "librosa 0.5.0," Feb. 2017.

[16] "Pescador," Retrieved January 2, 2018 from `https://github.com/pescadores/pescador`.

[17] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han, "Learning deconvolution network for semantic segmentation," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1520–1528.

[18] Yandre MG Costa, Luiz S Oliveira, and Carlos N Silla, "An evaluation of convolutional neural networks for music classification using spectrograms," *Applied Soft Computing*, vol. 52, pp. 28–38, 2017.

[19] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al., "Cnn architectures for large-scale audio classification," in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 131–135.

[20] Diederik Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[21] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio, "Why does unsupervised pre-training help deep learning?," *Journal of Machine Learning Research*, vol. 11, no. Feb, pp. 625–660, 2010.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2015, pp. 234–241.

[23] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang, "Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1874–1883.

[24] Yuting Zhang, Kibok Lee, and Honglak Lee, "Augmenting supervised neural networks with unsupervised objectives for large-scale image classification," in *International Conference on Machine Learning*, 2016, pp. 612–621.