Speech Command Classification using Dense-Recurrent Neural Network

EL-GY 9173: Audio Content Analysis
Project Report

Keerthi Sravan Ravi - N15874088
Aakaash Jois - N14182682

Department of Electrical and Computer Engineering
Tandon School of Engineering
New York University

1 Abstract

The commercial success of Amazon's Alexa and Google's Home products has been largely driven by successfully applying machine-learning techniques together with signal-processing concepts. The rise of machine-learning techniques have aided in achieving astounding accuracies in classifying speech even in noisy environments. In this work we compare multiple neural network architectures in classifying human-speech commands in nearly noise-free and partially noisy recordings. We test and compare implementations of vanilla convolutional neural networks, dense neural networks and dense-recurrent neural networks. We find that the dense network demonstrates the highest accuracy while the dense-recurrent network is most prone to overfitting on training data.

2 Introduction

Voice-driven interaction is being introduced, if not already present, in almost every form of human-computer interaction. Classifying human speech commands is one of the first steps involved when a user interacts with a digital assistant: the digital assistant has to analyze whether the user intended to activate the system or not. This has made classifying speech commands an extremely relevant and challenging problem in the current technological scenario. The difficulty is amplified when the user interacts with the system in a noisy environment.

Speech is a complex time-varying signal with complex correlations at a range of different timescales. Conventional deep-learning neural networks can provide only limited temporal modeling by operating on a fixed-size sliding window of acoustic frames. A novel approach at this classification task has been to apply convolutional neural networks (conventionally applied to tackle spatial-input problems) to feature maps produced from audio. These feature maps could be Mel-power spectrograms, CQTs, etc. Recurrent Neural Networks (RNN) have demonstrated greater success in the same tasks because they contain cyclic connections that enables them to more accurately model time-sequential data. The long short term memory architecture (LSTM [1]) overcomes some of the weaknesses of the simpler RNNs. Gated Recurrent Units (GRUs) are a gating mechanism in RNNs. GRUs have fewer parame-

ters than LSTMs, whilst demonstrating comparable performance on smaller datasets.

This work compares three different neural network implementations to classify human speech commands in English.

2.1 Dataset

The dataset is a part of Google Brain's TensorFlow Speech Recognition Challenge on Kaggle [2]. Released in August 2017, version 0.01 of the dataset contains 64,727 one second-long audio files. Six of these files contain only background noise. Each of the remaining one-second *.wav* audio files contain a single spoken English word. These words form a set of commands spoken by a variety of different speakers.

The speakers recorded twenty core commands, with most speakers saying each of them five times. The core words are "Yes", "No", "Up", "Down", "Left", "Right", "On", "Off", "Stop", "Go", "Zero", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", and "Nine". To help distinguish unrecognized words, there are also ten auxiliary words, which most speakers only said once. These include "Bed", "Bird", "Cat", "Dog", "Happy", "House", "Marvin", "Sheila", "Tree", and "Wow". The original audio files were collected in uncontrolled locations by people around the world.

The dataset includes two files that list the filenames for the validation and testing data splits. These text files contain the paths to all the files in each set, with each path on a new line. Any files that aren't in either of these lists can be considered to be part of the training set. However, the dataset is not speaker-conditional, because 1831 speakers recorded 64,721 files.

3 Methods

3.1 Data preprocessing

The Kaggle dataset contained audio recordings captured in a variety of formats, and then converted to a 16-bit WAVE file at 16000 sample rate. The dataset included two files that listed the filenames for the validation and test-

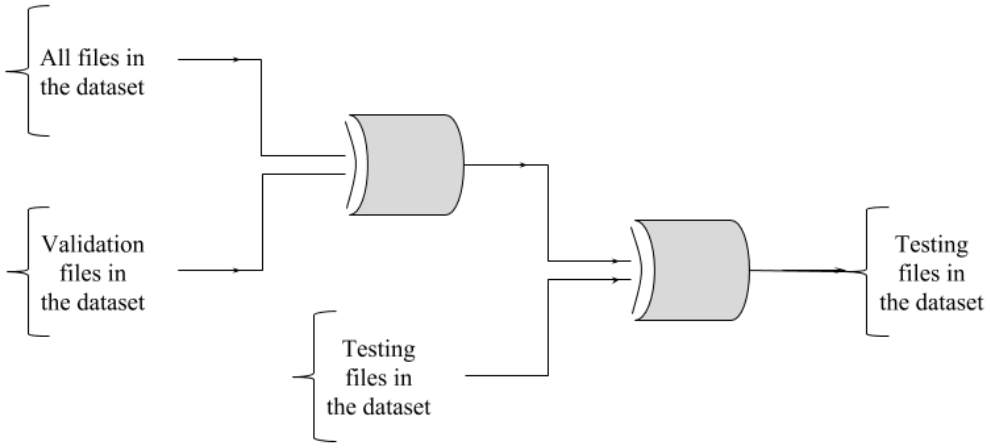

Figure 1: Making the training split

ing data splits. The training split was formed by performing a logical XOR operation on all files, validation files and testing files. This gave file names that were neither present in the validation split nor the testing split. Figure 1 illustrates the concept behind making the training split. The dataset could not be assumed to have exactly one-second long audio files, i.e. every audio file could not be expected to be exactly 16000 samples long. Each audio file shorter than 16000 samples was zero-padded equally at the beginning and end to be 16000 samples long.

A corrupted version of the dataset was generated, wherein each audio file was transformed with noise using the open-source Musical Data Augmentation (muda) library [3]. With a 50% probability, each audio file was transformed with random additive Gaussian white noise. The corrupted dataset was created to make each of the networks more robust by reducing over-fitting of training data.

Mel-power spectrograms were generated for each audio file (both clean and noisy) using Librosa [2] - an open-source music and audio analysis package. The parameters for generating the spectrograms were as follows: NFFT of 1024, and maximum frequency of 3kHz. The maximum frequency cut-off of 3kHz was chosen because human speech lies in the 500Hz - 1.5kHz range.

The Mel-power spectrograms were converted to the dB scale, and saved as single Numpy array.

3.2 Network architectures

Three different neural network architectures were implemented in Keras with Tensorflow backend [4]. A high-level description of the three architectures was as follows: the first architecture was a vanilla convolutional neural network, hereon referred to as the ConvNet. The second architecture incorporated a skip-layer (i.e. it was a dense network), hereon referred to as the DenseNet. It is important to note that the features from the previous layer were concatenated, and not added as done in a residual network. This technique of concatenating features from a previous layer was documented by Gao et al [5] to produce better results in classification than a residual network. The third architecture was based off the second architecture, and also implemented bi-directional recurrent layer at the end, hereon referred to as the Dense-Recurrent Net. A batch-normalization layer and a 20% dropout after every convolution layer was standard across all three architectures, except if a skip-layer was introduced (as in the case of the second and third architectures). Every convolution layer implemented kernels of size 8×3, and the ReLU activation function. A 'taller' convolutional kernel was chosen because the Mel-power spectrogram had a greater spread of spectral information in the y axis per time-point, and it was important to include this information in each convolution operation. Figure 2 illustrates the three network architectures.

All three networks were trained on both the clean and noisy datasets for 100 epochs, with a batch-size of 512. All three networks implemented Keras callbacks for decreasing the learning-rate on plateauing, saving the best model at every check-point and early-stopping of training.

4 Results and discussion

K-fold cross-validation was not performed since the dataset included a predetermined split. The Kaggle dataset comprised of 6469 files containing fewer than 16000 samples. Each log Mel-power spectrogram was a matrix of size

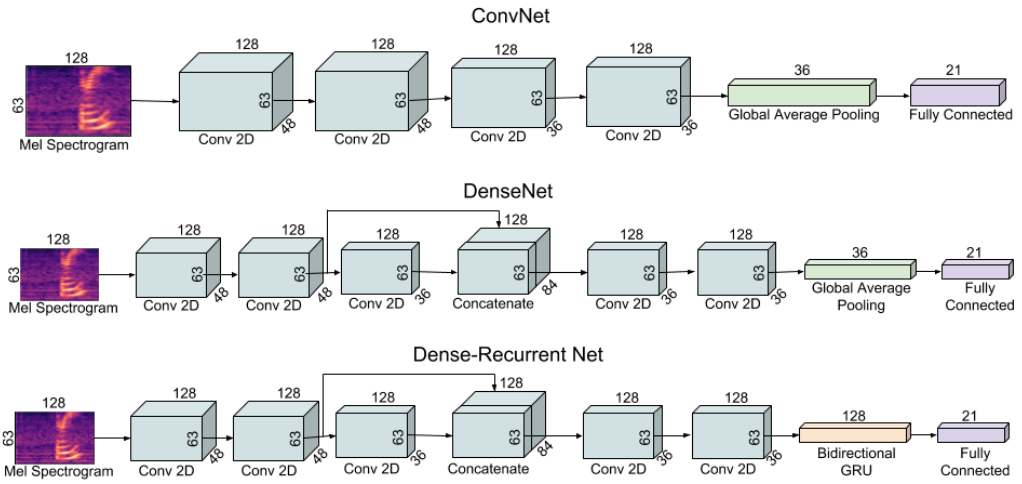

Figure 2: Model architectures

128×63. It is important to note that the clean dataset was transformed in muda using a single Gaussian noise sample, with a binary weighting. That is, each audio file was transformed by noise with a 50% probability.

It is supposed that if the models were trained on time-domain representations, they would have learned the waveform of the noise and would have tended to overfit. In this work however, the models were trained on frequency-domain representations.

Table 1 lists the accuracies of all three architectures on both the clean and noisy datasets. In both cases, with the early-stopping mechanism in place, the ConvNet took the largest number of epochs to train, followed by the DenseNet and the Dense-Recurrent Net respectively. All three models performed better on the clean dataset. The ConvNet's accuracies decreased on average by 7.56% when trained on the noisy dataset. Whereas, the DenseNet's and Dense-Recurrent Net's accuracies decreased on average by 2.33% and 1.22% respectively when trained on the noisy dataset. The performances of all three models on the noisy datasets are discussed, unless mentioned otherwise.

Dataset	Accuracy	ConvNet	DenseNet	Dense-Recurrent Net
Clean	Train	85.68%	90.28%	99.71%
	Validation	80.47%	84.74%	83.45%
	Test	81.49%	85.04%	83.39%
Noisy	Train	88.02%	88.67%	99.59%
	Validation	83.64%	81.95%	82.74%
	Test	84.14%	83.20%	82.60%

Table 1: Accuracies of all three architectures on the clean and noisy datasets.

The Dense-Recurrent Net in particular was prone to over-fitting on the training data. This can be clearly observed by comparing the training and testing accuracies of 99.59% and 82.60% . This could be attributed to the bi-directional GRUs attempting to learn to memory the limited temporal sequencing information available.

Figure 4 is a heat-map of the confusion matrix of the ground truths and the predictions of the Dense-Recurrent Net. Certain 'unknown' commands were often incorrectly classified. Figure 4 is a heat-map of these incorrect classifications: it illustrates which 'unknown' command in particular was incorrectly classified as a valid command. For example, 'tree' is most often incorrectly classified as 'three'. Figure 5 is a comparison of the spectrograms for 'three' and 'tree' recorded by the same speaker, and the reason for this incorrect classification is more understandable.

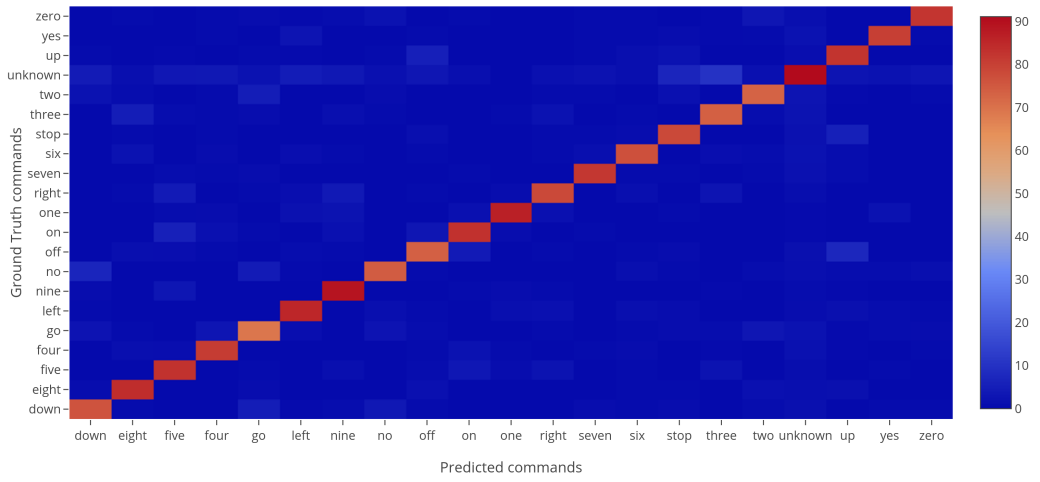

Figure 3: Heat map of predictions by the Dense-Recurrent Net.

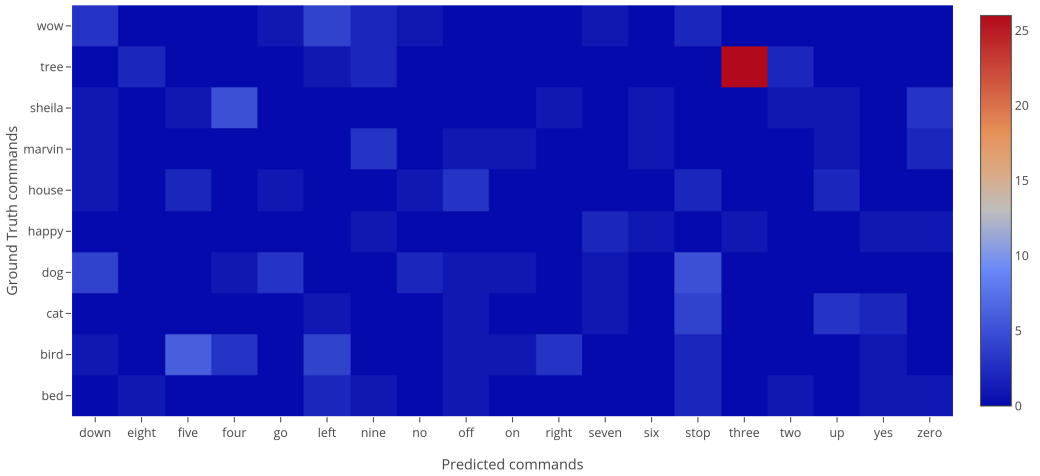

Figure 4: Heat map of incorrect 'unknown' command predictions by the Dense-Recurrent Net.

Figure 6 shows a plot of the standard deviations on the command classifications for all three models. It can be observed that the Dense-Recurrent Net has a narrower spread. That is, even the worst-performing categories on the Dense-Recurrent Net will perform closer to the reported mean accuracy. This is not the case with the other two models, which have a wider spread by measure of standard deviation.

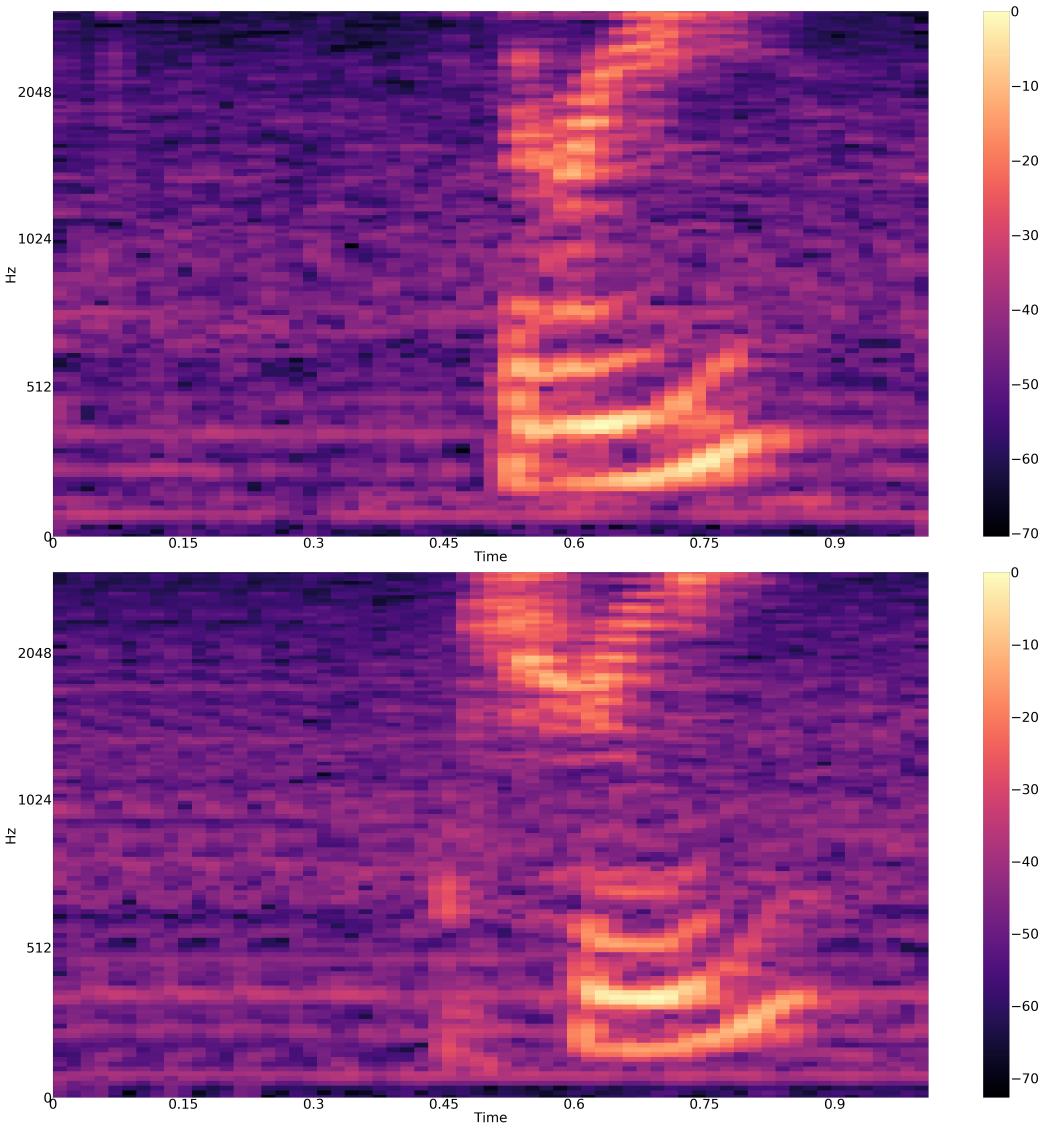

Figure 5: Spectrograms of 'three' and 'tree', recorded by the same speaker.

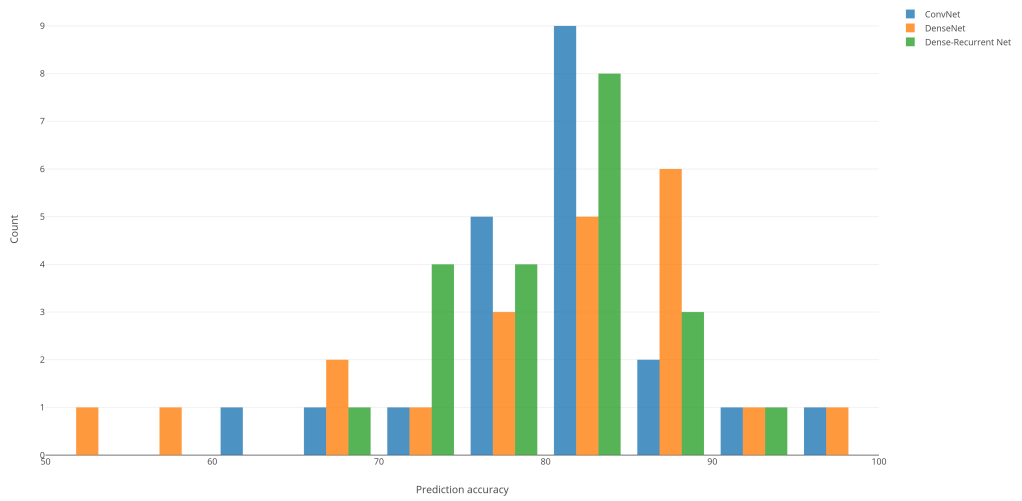

Figure 6: Plot of standard deviations for all three models on command classification accuracies.

It is important to note that it is not possible to determine which of the speech commands were incorrectly classified as which 'unknown' command in particular. This is because this information is inherently lost during the training of the model.

The Keras callback implemented to save the best model largely prevented overfitting on the training data. All the models stopped training before the stipulated 100 epochs were completed. Figure 7 depicts a graph of progression of training and validation accuracies as a function of the number of epochs.

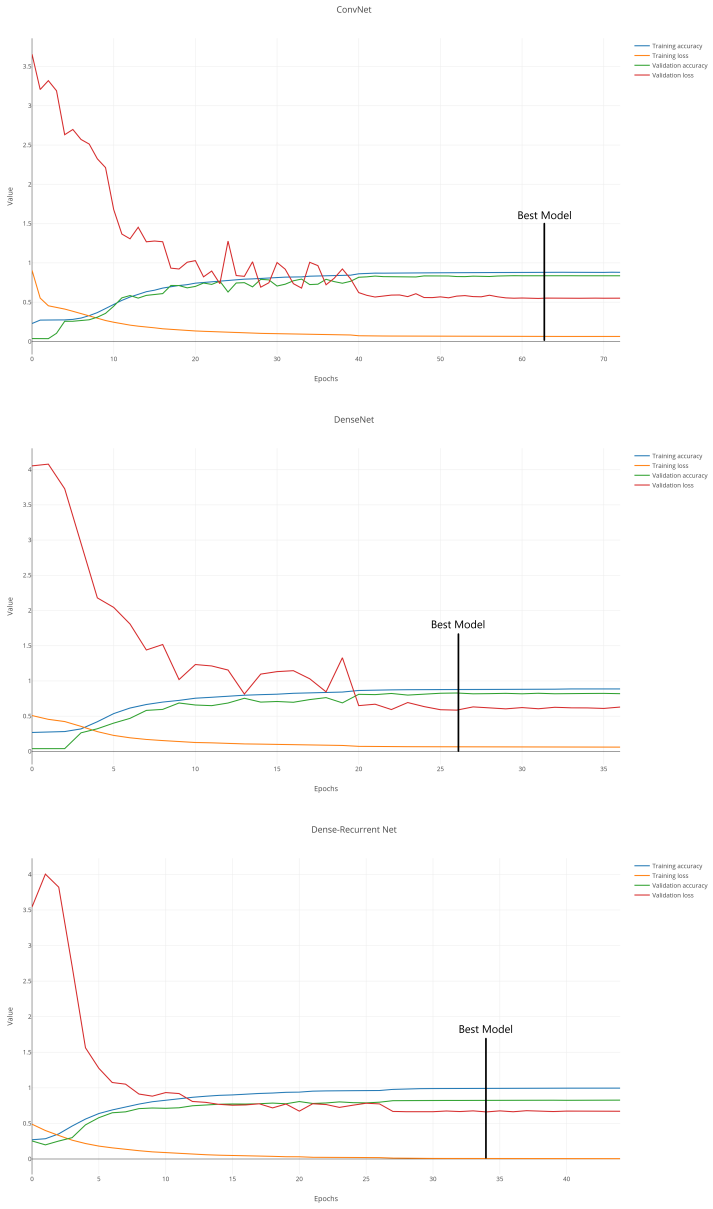

Figure 7: Graphs of training and validation accuracies as a function of number of epochs. The black line indicates the epoch of each model's best performance.

5 Conclusion

Training the ConvNet on the noisy dataset made it more robust, as observed by in the increase in testing accuracy. The DenseNet and Dense-Recurrent Net are also robust to noise. However the Dense-Recurrent Net tends to overfit, as can be observed by the wide gap in training and testing accuracies in table 1. Based on figure 6 it could be concluded that the Dense-Recurrent Net is the best model, as substantiated by the reasons listed above.

This work validates that a variety of end-to-end deep learning architectures can be used reliably to classify human speech commands. No matter the architecture utilized, a comprehensive evaluation metric is important to convey a realistic expectation of the model's performance.

References

[1] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Fifteenth annual conference of the international speech communication association*, 2014.

[2] P. Warden, "Speech commands a public dataset for single-word speech recognition.," *Dataset available from* `http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz`, 2017.

[3] B. McFee, E. J. Humphrey, and J. P. Bello, "A software framework for musical data augmentation.," in *ISMIR*, pp. 248–254, Citeseer, 2015.

[4] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

[5] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks," *CoRR*, vol. abs/1608.06993, 2016.