ARCADA

Single-word speech recognition with Convolutional Neural Networks on raw waveforms

Patrick Jansson

Degree Thesis

Information technology

2018

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informationsteknik
Identifikationsnummer:	6603
Författare:	Patrick Jansson
Arbetets namn:	Taligenkänning av enskilda ord med Konvolutionella neuronnät på rå ljuddata
Handledare (Arcada):	Shuhua Liu
Uppdragsgivare:	

Sammandrag:
Examensarbetet fokuserar på taligenkänning av enskilda ord med ett konvolutionellt neuronnät, där målet är att lära sig klassificera ett antal fördefinierade ord. Taligenkänning av enskilda ord kan användas som röstgränssnitt med nyckelordsigenkänning på t.ex. smarttelefoner och inbäddade system. Dessa apparater har ofta strikta krav för datakraft och minne, vilket beaktas i examensarbetet fastän det huvudsakliga målet inte är en modell som kan användas i dem. Examensarbetet använder sig av datamängden Speech Commands, som består av ca 65 000 en sekund långa uttalanden med 30 engelskspråkiga ord som t.ex. ”left”, ”right” och ”stop”. Målet är att klassificera 10 av orden i datamängden, samt två separata klasser för alla andra ord och tystnad. För att klassificera orden används ett konvolutionellt neuronnät, som består av endimensionella konvolutioner för att användas på rå ljuddata. Målet är att utnyttja kraften av djupinlärning där neuronnätet självt lär sig särdragsrepresentationer under inlärningprocessen. Traditionella metoder för taligenkänning använder sig däremot av funktionsteknik för att förvandla dataexemplaren till exempelvis spektrogram, som sedan används av modellen. För effektiv inlärning används fyra huvudsakliga metoder: viktad slumpmässig provtagning, Stochastic Gradient Descent with Warm Restarts (SGDR), artificiell utvidgning av datamängden och användning av pseudo-etiketter. Viktad slumpmässig provtagning används för att jämna ut klassdistributionen under inlärningsprocessen. SGDR är en metod för hur man ändrar inlärningstakten efter varje inlärningsiteration. Artificiell utvidgning av datamängden består av att man modifierar de tillgängliga dataexemplaren på ett sätt där de fortfarande kan identifieras med samma etikett. För att använda sig av pseudo-etiketter utnyttjar man en färdigt inlärd modell som gör förutsägelser på okänd data. Dessa förutsägelser används sedan som sanna värden för inlärning av en ny modell. Den slutliga modellen uppnår 97,4% noggrannhet på valideringsdatamängden, och 88,7% samt 89,4% noggrannhet på två testdatamängder. Resultaten visar att modellen med hög noggrannhet kan förutsäga ord som den sett under inlärningprocessen, men det uppstår vissa problem med att förutspå nya ord samt exemplar med mycket backgrundsoljud eller låg ljudkvalitet.

Nyckelord:	konvolutionella neuronnät, taligenkänning, maskininlärning, djupinlärning
Sidantal:	27
Språk:	Engelska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information technology
Identification number:	6603
Author:	Patrick Jansson
Title:	Single-word speech recognition with Convolutional Neural Networks on raw waveforms
Supervisor (Arcada):	Shuhua Liu
Commissioned by:	

Abstract:
This work focuses on single-word speech recognition, where the end goal is to accurately recognize a set of predefined words from short audio clips. It uses the Speech Commands dataset, which consists of 65 000 one-second long utterances of 30 short words of which we learn to classify 10 words, along with classes for "unknown" words as well as "silence". Single-word speech recognition can be used in voice interfaces for applications with keyword detection, which can be useful on mobile and embedded devices. These devices often have strict requirements in terms of computing power and memory, which is recognized in the design of the speech recognition model. To classify samples, we use a Convolutional Neural Network (CNN) with one-dimensional convolutions on the raw audio waveform. As opposed to more traditional methods where feature-engineering is crucial, we leverage the power of deep learning to learn the feature representation during training. To effectively train the model we use data augmentation, weighted random sampling, Stochastic Gradient Descent with Warm Restarts (SGDR) and pseudo-labeling. The model achieves 97.4% accuracy on the validation set, 88.7% and 89.4% on the two test sets. The results show that the model can predict samples of words it has seen during training with high accuracy, but it somewhat struggles to generalize to words outside of the scope of the training data and extremely noisy samples.

Keywords:	deep learning, neural network, convolutional neural network, speech recognition, keyword spotting, artificial intelligence
Number of pages:	27
Language:	English
Date of acceptance:	

CONTENTS

Figures

Tables

1 INTRODUCTION

Deep learning and neural networks have achieved state of the art results in various tasks such as image classification (Simonyan, Zisserman 2014), machine translation (Wu et al. 2016) and speech recognition (Amodei et al. 2016). These neural network models can consist of up to tens of millions of parameters where both training and inference require a lot of time and powerful hardware. While these models achieve state of the art results, they are not able to be efficiently used on devices such as mobile phones and embedded devices. Model architectures like MobileNet by Howard et al. (2017) have been created for efficient inference and low memory usage while still achieving comparable results to the state of the art.

This work focuses on single-word speech recognition or keyword spotting (KWS), where the end goal is to accurately recognize a set of predefined words from short audio clips. Zhang et al. (2017) discussed the importance of high accuracy and low compute requirement models needed for efficient use of keyword spotting on devices like mobile phones and microcontrollers. They pointed out problems with always-on, over the network, speech recognition on devices like Amazon Echo, Google Home and smart phones, where transmitting continuous streams of audio is not only inefficient, but also leads to privacy concerns. They also discussed how some of these problems are mitigated using KWS, where the device is only listening for specific keywords such as "Ok Google", and when recognized it enables the full-scale speech recognition system. This allows the device to use more complex models only when needed and alleviates some of the privacy concerns.

The focus of this work is on the use of one-dimensional convolutions on raw audio data as well as effective training methods, to create a model which achieves the highest possible accuracy. This work was done as a part of the TensorFlow Speech Recognition Challenge[1] on Kaggle, a platform for data science and machine learning competitions. The goal of the competition was to achieve the highest possible accuracy on the test set provided by Kaggle, where the labels for the set were not given out and results are given based on prediction submissions done through the Kaggle website.

While the main use case of models like these are on smart-devices, this work doesn't specifically focus on that use case, but it does acknowledge it both in terms of design of

[1] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge

the model architecture as well as results. Real-life use cases generally also require classification from a continuous audio stream, this work doesn't address this problem as the main focus is on classifying the test set data, which only requires one prediction per sample.

This work was done while working on the Katumetro[2] research project on intelligent methods and models for mining community knowledge from social media.

1.1 Speech recognition

Although single-word speech recognition differs a lot from full scale speech recognition, many of the underlying ideas are the same. Traditional speech recognition systems commonly use Hidden Markov models (HMMs) along with a lot of feature engineering and Gaussian mixture models (GMMs) for acoustic models. The first steps towards using neural networks in speech recognition were using neural networks for acoustic modeling instead of GMMs. (LeCun, Bengio & Hinton 2015) These have since been mostly replaced by end-to-end trained neural architectures such as Deep Speech (Hannun et al. 2014, Amodei et al. 2016).

1.2 Deep learning

Deep learning is a subset of machine learning where the models, generally artificial neural networks, use multiple layers to create a richer representation of the data. Traditional machine learning algorithms like support vector machines, require hand-made features to reach optimal results, whereas deep learning algorithms are capable of creating their own features from more raw data, based on what is learned to be relevant during training.

> Deep learning allows computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. These methods have dramatically improved the state-of-the-art in speech recognition, visual object recognition, object detection and many other domains such as drug discovery and genomics. Deep learning discovers intricate structure in large data sets by using the backpropagation algorithm to indicate how a machine should change its internal parameters that are used to compute the representation in each layer from the representation in the previous layer. (LeCun, Bengio & Hinton 2015)

[2] http://rdi.arcada.fi/katumetro-digilens-hki/en/

This thesis focuses on both supervised and semi-supervised learning. Supervised learning is the most common form of machine learning. Supervision entails that training leverages labeled data to teach the algorithms. In this work we train a deep neural network to recognize keywords such as “yes” or “no” from short utterances where the label for each utterance is the keyword. Semi-supervised learning is very similar to supervised learning, but it tries to leverage unlabeled data along with the labeled data. Training neural networks is done using Stochastic Gradient Descent (SGD) or some of its variants like Adam (Kingma, Ba 2014). SGD uses the error between the predictions of the model and the labels to adjust the weights so that the error is minimized. This process is done in batches of training examples and repeated until convergence. (LeCun, Bengio & Hinton 2015)

The basic neural network architecture, or the feedforward neural network consists of fully-connected layers and non-linear functions where a weighted sum of the inputs to a fully-connected layer is computed and passed through a non-linear function. Commonly used non-linear activation functions are sigmoid and hyperbolic tangent (tanh), as well as the Rectified Linear Unit (ReLU) which has largely replaced the other activation functions as they show improvements in training deep networks (Glorot, Bordes & Bengio 2011). The output of the final layer is often passed through a softmax function so that each output is mapped to a probability representing the predictions for each class in the supervised learning setting (LeCun, Bengio & Hinton 2015). While the advancements in deep learning and neural networks are centralized in the current decade, neural networks have been around since the 1940s (Schmidhuber 2015).

2 THEORY AND METHODS

This section introduces the basic theory of how to use audio data for speech recognition models, the building blocks used in the final model architecture as well as the basic principles behind training neural networks.

2.1 Sound as data

The dataset consists of a set of WAVE-files, which are all roughly one second long. To use the data, each file is sampled into a vector with a sampling rate of 16000. A common strategy for speech recognition is to first extract features from the raw waveform. Commonly used speech features like spectrograms, log-Mel filter banks and Mel-frequency cepstral coefficients (MFCC) convert the raw waveform into a time-frequency domain (Zhang et al. 2017). These features are then used as an input to a model. Sainath and Paranda (2015) show how log-Mel filter banks can be used as input features to a neural network.

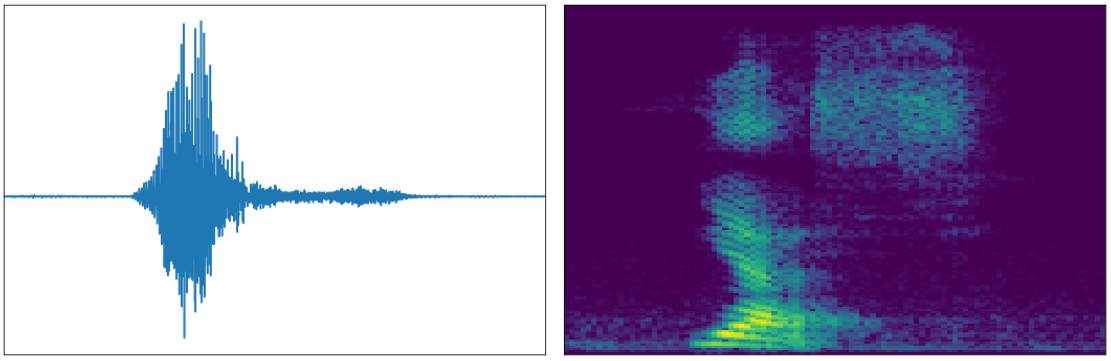

Figure 1. A sample depicting the word "yes" as the raw waveform (left) and as a log-spectrogram (right)

Dai et al. (2017) discuss the challenges with audio-feature-engineering, which requires certain domain knowledge without necessarily building optimal features. Instead of using feature engineering of any kind, this work aims to leverage the power of deep learning to discover features during training from the raw waveform.

2.2 Convolutional neural networks

Convolutional neural networks (CNN), originally introduced by LeCun et al. (1990, 1998) were popularized by AlexNet after showing significant improvements over other

submission in the ImageNet ILSVRC challenge in 2012 (Krizhevsky, Sutskever & Hinton 2012). Since then, Convolutional neural networks have achieved state of the art results in various task in computer vision such as object-detection (Redmon et al. 2016) and image segmentation (Badrinarayanan, Kendall & Cipolla 2017), as well as machine translation (Wu et al. 2016), text classification (Kim 2014) and speech recognition (Amodei et al. 2016).

> Deep neural networks exploit the property that many natural signals are compositional hierarchies, in which higher-level features are obtained by composing lower-level ones. In images, local combinations of edges form motifs, motifs assemble into parts, and parts form objects. Similar hierarchies exist in speech and text from sounds to phones, phonemes, syllables, words and sentences. (LeCun, Bengio & Hinton 2015)

Convolutional neural networks consist of four main operations: convolutions, non-linearities, pooling and classification. Optionally the models can include batch normalization as well as dropout. These operations are commonly stacked together so that convolutions are followed by a non-linearity such as a ReLU, this operation is then repeated a few times after which a pooling operation is used. When the network is deep enough, and the original input is subsampled by pooling to a size where it is manageable, it is passed through to the classification part of the network. If batch normalization is applied, it's commonly used after the convolution but before the non-linearity.

2.2.1 Convolutional layers

Convolutional layers in a CNN consists of a set of learnable filters, sometimes also called kernels. Each filter is applied by independently striding over the entire input, creating an output feature map for each filter. Applying the same filter over the entire input space leverages the fact that the same motif can appear in different locations in the input. Convolutional layers often use a relatively small filter sizes, which allows detection of local features which for data such as images and sound are crucial. Filters in stacked convolutional layers operate on the feature map from the previous layer, combining features to detect higher order features. (LeCun, Bengio & Hinton 2015)

The most common convolutional layer operates on 2-dimensional inputs, the height and width of images for instance. Since this work uses the 1-dimensional raw audio waveform data as input, the convolutional layers use 1-dimensional filters.

2.2.2 Pooling layers

While convolutional layers detect local features from its input, a pooling layer merges semantically similar features by only keeping the maximum value (max-pooling) or averaging (average-pooling) the values within a patch (LeCun, Bengio & Hinton 2015). This reduces sensitivity to shifts and distortions of features, by reducing the size of the feature map. Pooling layers are used after one or a few convolutional layers, continuously reducing the size of the input as the network gets deeper. Every reduction in the input size followed by the pooling operation also reduces the amount of computation needed in the network. (LeCun et al. 1998)

While most modern convolutional neural networks use max-pooling layers, some argue against the use of pooling layers as they simply discard potentially valuable information. Springenberg et al. (2015) show that replacing max-pooling layers with convolutions using a larger stride to reduce the input size can yield competitive results compared to models using max-pooling.

2.2.3 Batch normalization

As a neural network is trained and the networks parameters are updated, the distribution of weight activations change, this is referred to as internal covariate shift. Internal covariate shift makes training much more difficult, by requiring lower learning rates and careful parameter initializations. To address this problem, Ioffe & Szegedy (2015) introduced batch normalization, which makes normalization a part of the model itself. Batch normalization normalizes the output from a previous layer by using estimates of the mean and variance on a batch-level. Batch normalization also introduces two trainable parameters, gamma and beta, which are used to scale and shift the normalized values restoring representation power of the network. Batch normalization allows the use of higher learning rates, as the normalization process addresses issues with vanishing and exploding gradients, stabilizing the training process. During inference, instead of per batch statistics, mean and variance statistics are used from the whole training data. (Ioffe, Szegedy 2015)

2.2.4 Activation functions

Activation functions are non-linearities used between convolutional layers so that the neural network can model more complex than linear data. A common activation function is the Rectified Linear Unit (ReLU), which is a function that takes input x and returns *max(0, x)*. The ReLU and variants of it such as PReLU have mostly replaced the older activation functions sigmoid and tanh. This is due to much greater training efficiency compared to the older activation functions. (Krizhevsky, Sutskever & Hinton 2012, Glorot, Bordes & Bengio 2011)

2.2.5 Classification

In most cases the classification part of convolutional neural networks consists of fully-connected layers and a softmax function. Fully-connected layers are used after the final convolutional layer in order to match the output size of the neural network to the desired output size. The output is then passed through a softmax function in order to create a probability representation for the predictions for each class in the supervised learning setting. To use the fully-connected layers, the output from the final convolutional layer is commonly flattened out, or the feature maps are subsampled to a size of 1.

2.2.6 Dropout

Dropout is a regularization technique where a percentage of connections between units in a layer are dropped before the following layer. Randomly dropping a set of units for each training iteration prevents co-adaptation by making the presence of other units in the network unreliable. This forces the network to learn a more general representation, rather than learning specific connections. A downside with using dropout is increased training time as the broken connections increase noise in parameter updated. No connections are dropped during inference. (Srivastava et al. 2014)

2.3 Training deep neural networks

Neural networks are trained using backpropagation and gradient descent. Backpropagation is used to compute the gradients in respect to each weight in the network based on the error of the output. In classification tasks, the error is commonly the cross-entropy loss between the output of the network and the target. The gradients are used to give a direction in which to adjust the weights so that the error is minimized. Computing the outputs and the error for a batch of inputs, computing the average gradient for the examples in the batch and adjusting the weights based on the gradient is called Stochastic Gradient Descent (SGD). Stochastic Gradient Descent is a stochastic approximation of gradient descent optimization, where instead of taking a single training step for each full iteration of the training set, steps are taken after each batch where the per-batch average gradient acts as an estimate for the true gradient. (LeCun, Bengio & Hinton 2015)

The amount which the weights are updated by is defined by the gradient as well as a learning rate. This value plays a large part in training neural networks, as a learning rate which is too high will never reach a proper minimum and a learning rate which is too low can get stuck is suboptimal minima or require very long training times as each training step only updates the weight by a small margin. To counter this, it's common to use some sort of learning rate schedule such as turning down the learning rate after a certain number of updates, or when the loss on a held-out validation set has stopped decreasing (Smith 2015). Another option is to use optimizers like Adam (Kingma, Ba 2014) or RMSProp (Tieleman & Hinton, 2012) that use adaptive learning rates.

3 DATA, TOOLS AND PRE-EXPERIMENTS

The work in this thesis is implemented in Python[3]. The main Python library used in this work is PyTorch (Paszke et al. 2017), which is a deep learning framework that allows its users to create and train powerful deep learning models, with possibility to do computations on a Graphics Processing Unit (GPU). The other main libraries used are NumPy (Oliphant 2006), Pandas (McKinney 2010), scikit-learn (Pedregosa et al. 2011), and Librosa (McFee et al. 2017).

3.1 Dataset

The availability of large public datasets such as ImageNet (Deng et al. 2009) for image classification and Microsoft COCO (Lin et al. 2014) for object detection, segmentation and captioning have accelerated the advancements in deep learning.

This thesis uses the Speech Commands dataset (Warden 2017b). The dataset consists of 65 000 one-second long utterances of 30 short words such as “yes”, “no”, “right” and “left”. The dataset aims to help with building voice interfaces for applications with keyword detection, which can be useful on mobile devices and microcontrollers (Warden 2017a).

The goal is to classify the following words: "yes", "no", "up", "down", "left", "right", "on", "off", "stop" and "go". All other words are labeled as “unknown” and are used to help the model learn a representation for all words which are not in the 10 words to be classified. The final class is “silence”, which represents samples with no word. The test

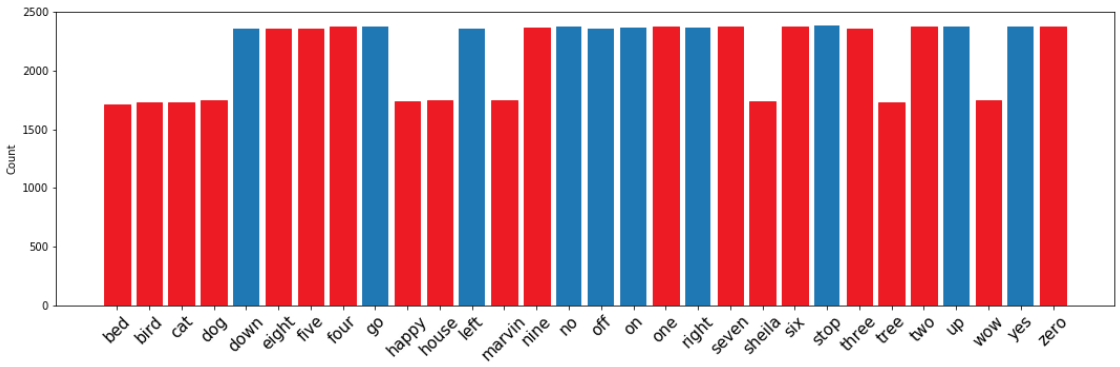

Figure 2. Counts for each class in the Speech Commands dataset. Blue examples represent words that should be classified, red examples represent the unknown class

[3] Code available on GitHub: https://github.com/PJansson/speech

set provided by Kaggle consists of roughly 150 000 additional utterances, including words and speakers that do not appear in the Speech Commands dataset.

The files in the dataset are arranged into folders by label, the filenames start with a hash representing the speaker, followed by a number representing the number of times that utterance by the same speaker appears in the dataset. The data is split into training and validation sets using the hash so that the same speaker does not appear in both sets.

Kaggle also provided a set of audio files which are denoted as background noise, using these files we create data for the silence class as well as background noise which is used during training. Some files which were found to be silence but were mislabeled were corrected and used as additional silence during training.

3.2 Pre-experiments

To reach the final implementation, including model architecture, data augmentation techniques and training setup, an iterative process of experiments were covered. These experiments included testing different model architectures, mainly inspired by VGGNet (Simonyan, Zisserman 2014), ResNet (He et al. 2016) and DenseNet (Huang et al. 2017), all modified to use 1-dimensional convolutions. Along model architectures and hyperparameters various training- and data augmentation-techniques were also experimented with. The final implementation is described in detail in Section 4, a discussion on the results of the experiments can be found in Section 6.1.

4 IMPLEMENTATION SPECIFICATION

4.1 Model architecture

The final model architecture is heavily inspired by VGGNet (Simonyan, Zisserman 2014) and the models used by Dai et al. (2017). The model consists of six blocks, where each block contains two convolutional layers, which are both followed by batch normalization layers as well as ReLU-activation functions. The convolutional layers use one-dimensional convolutions with a kernel size of 5, unlike the work by Dai et al. (2017), the first layer uses a smaller kernel size which is then kept throughout the network. The number of convolutional filters for the first block is 8, after which the number is doubled for every block ending with 256 filters. Each block is followed by a pooling operation, all but the final pooling layers use max-pooling, the last pooling layer is a global average pooling layer which takes each feature map and returns the average, each max-pooling layer uses a kernel-size of 4.

The global average pooling layer is followed by the classification part of the network, which consists of two fully-connected hidden layers and the output layer. The output sizes of the hidden layers are 128 and 64, both hidden layers are followed by ReLU-activations and dropout with a rate of 0.5. The output layer has an output size of 12, to match the number of classes in the dataset. The total amount of parameters in the network is roughly 0.7 million.

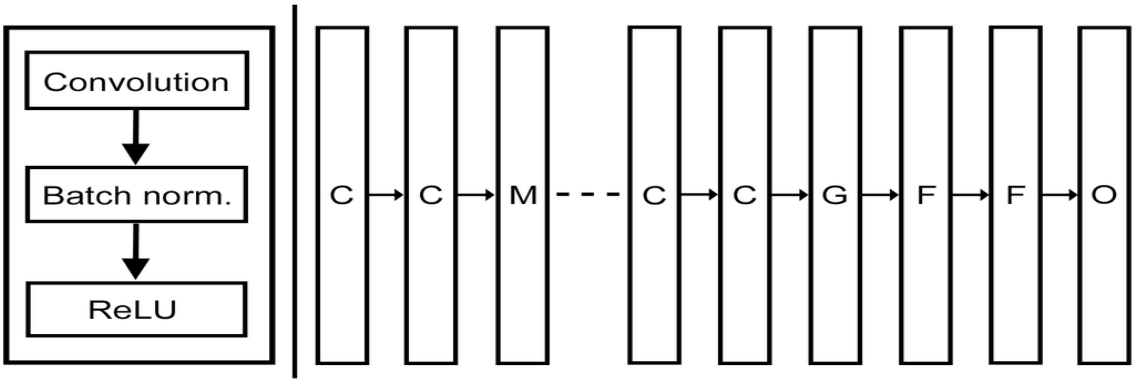

Figure 3. Model architecture (right), middle layers omitted for clarity. C = conv-batch norm-relu (left), M = max-pooling layer, G = global average pooling layer, F = fully-connected layer, O = output layer

4.2 Model training

4.2.1 Data augmentation

Data augmentation is a way to increase the amount of training data by modifying the available data in a way where it can still be identified with the same label. Data augmentation has shown to be a simple and effective way of reducing overfitting, and thus improving model performance. Data augmentation can also help the model learn a wider range of features as the augmented samples can be quite different from the original training samples while still being identifiable as the same. (Krizhevsky, Sutskever & Hinton 2012)

Three data augmentation techniques were used: shifting the audio in time, scaling the amplitude and adding noise. The first step, shifting in time, is applied with a 50% probability to each sample. The augmented samples are shifted forward or backward in time by up to 20% of the samples original length. Occasionally this causes the utterance in the sample to be partially cut out, but the probability of this happening is negligible. This augmentation technique should help the model learn a more time-invariant representation of the utterances, as they can appear anywhere within the sample.

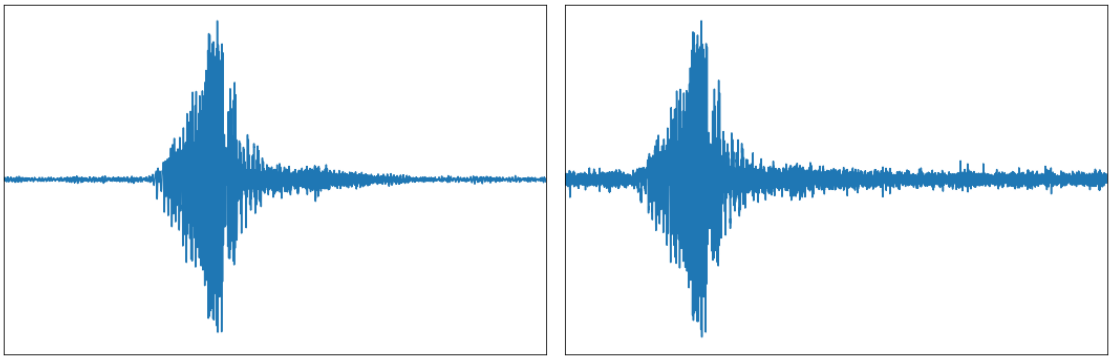

Figure 4. A sample of the word "yes" as the original sample (left) and an augmented version of it (right)

The second augmentation technique is mixing in noise with the sample, this is also applied with a 50% probability. Noise is generated in the same way silence is generated during training, where up to two samples from the dataset background audio are scaled down and added together. This noise is then added to the original input, which has been scaled between 75% and 125% of its original amplitude. Adding noise should help the model to better distinguish relevant information from the data.

4.2.2 Sampling

The training set is heavily skewed towards the unknown class, as shown in Figure 2. Training with the dataset as-is can lead to problems where the overrepresented class is favorably predicted. To combat the class imbalance during training, we use weighted random sampling with replacement, where instead of going through each sample in the training data for each epoch, each sample has a weight which equals to 1 / the number of the samples label in the training set. When selecting a sample to train on, the sample weight defines the probability of that sample to be chosen, leading to the more common classes to be down-sampled and the less common classes to be up-sampled. This leads to training on batches where each label is represented equally, but where all samples are still seen during training.

The usual definition of an epoch is iterating through the training set once, but because we are using sampling, an epoch is referred to as selecting samples until the amount is equal to the size of the training data.

4.2.3 Pseudo-labeling

Pseudo-labeling is a simple but effective semi-supervised training method, where a trained model is used to predict unlabeled data, in our case the test set which we don't have the labels to, then use the model predictions as pseudo-labels. The model is then re-trained using a combination of data from the original training set and pseudo-labeled data. The end goal of pseudo-labeling is to improve generalization performance by using the unlabeled data. (Lee 2013)

Pseudo-labeling introduces label noise into the training data since it's impossible to know if all the pseudo-labels are correct. Fortunately, deep neural networks can be trained on data where a significant percentage is miss-labeled as shown by Sukhbaatar and Fergus (2014). They show that with a large enough training set the effect of label noise isn't significant enough to warrant not using all the data available.

We use the predictions of the initial model for pseudo-labels only if the maximum output of the model, after going through a softmax function is higher than 0.95. This discards the data when the model prediction isn't very confident, theoretically decreasing label noise. Since we know that our test set contains utterances which are not included in, using pseudo-labels should allow our model to learn a better representation for the "unknown"

class. Once the pseudo-labels are selected, each sample is given a sample weight similarly as during the original training process. The model is then re-trained using both the original training data and the pseudo-labels so that 2/3 of each batch consists of samples from the original training data and 1/3 consists of samples from the pseudo-labeled data.

4.2.4 Learning rate schedule

A good learning rate schedule is crucial for not only reaching better results but also to possibly improve training time. Common learning rate schedules include turning down the learning rate after a certain number of updates, when the loss on a held-out validation set has stopped decreasing or linearly turning down the learning rate. Loshchilov and Hutter (2016) propose a training method they call Stochastic Gradient Descent with Warm Restarts (SGDR). SGDR introduces the concept of warm restarts by first decaying the learning with cosine annealing for each training iteration, followed by resetting the learning rate and repeating the cosine annealing process. Loshchilov and Hutter (2016) also show that using SGDR can result in competitive results in two to four times faster training time.

In this work we use SGDR with an initial learning rate of 0.1, which is trained for 70 epochs where the first restart is done after 10 epochs, and the epoch count for each consecutive cycle is doubled. The learning rate for the training process is shown in Figure 5 below, an additional cycle is added when training with pseudo-labels.

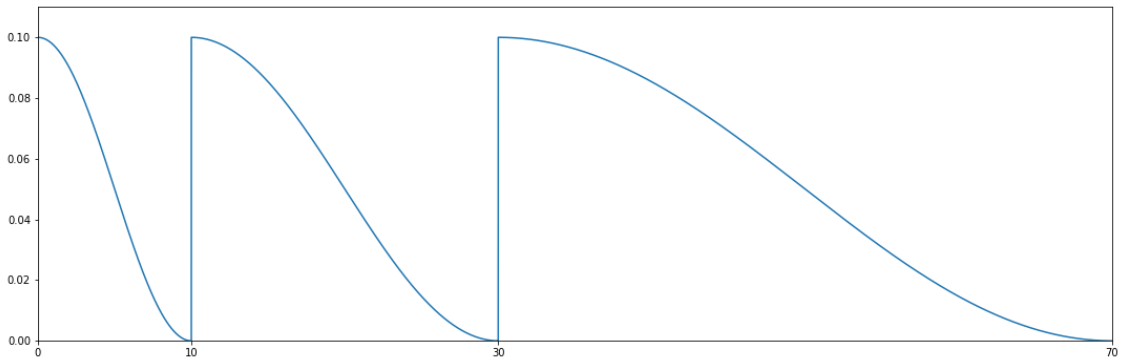

Figure 5. Learning rate schedule

5 RESULTS

The evaluation of results is split into three sets, the validation set as well as two test sets. The validation set contains roughly 10% of the samples in the dataset. The two test sets are based on the two Kaggle leaderboards, the public and the private leaderboard which contain 30% and 70% of the test data respectively, for clarity these will be denoted as the public test set and private test set. Accuracy, precision, recall and F1-scores are reported for the validation data, but as we don't have access to the correct labels for the test sets, only accuracy scores will be reported for them. Some analysis of manually reviewed predictions on the test set will be discussed, as the test sets contain words and speakers not in the training data and reviewing predictions will give greater insight to the strengths and weaknesses of the model.

Before the use of pseudo-labels in the training process the model achieves 97.5% accuracy on the validation set, 87.8% accuracy on the public test set and 88.4% accuracy on the private test set. After incorporating pseudo-labels into the training process, the model achieves 97.4% accuracy on the validation set, 88.7% accuracy on the public test set and 89.4% accuracy on the private test set.

Per class precision, recall and F1-scores for the validation set are reported in Table 1.

Table 1. Precision, recall, F-measure and support for validation set

Label	Precision	Recall	F1-score	Support
Yes	0,99	0,99	0,99	238
No	0,95	0,98	0,96	233
Right	0,94	0,97	0,95	258
Left	0,97	0,98	0,98	240
Up	0,97	0,98	0,97	224
Down	0,95	0,97	0,96	234
Go	0,88	0,98	0,93	211
Stop	0,94	0,97	0,95	205
On	0,93	1,00	0,96	205
Off	0,95	0,98	0,97	218
Silence	0,65	0,87	0,75	63
Unknown	0,99	0,97	0,98	3946

5.1 Discussion

The results show that the model is able to correctly predict a very high percentage of samples, however there is a significant difference between validation and test set accuracies. Manually reviewing test set predictions show that the test set is much more difficult than the validation set. It contains much more noise in the samples, along with a large number of samples which are cut out in such a way that the complete word cannot be heard. This causes difficulties for the model, as in many of the samples it is very hard to make out an utterance, let alone a specific word. Another reason is the addition of completely new words, which are occasionally predicted as one of the 10 distinct words in the training set as opposed to the unknown class, two words which are often predicted incorrectly are “backward” and “learn”. The word “learn” is most commonly predicted as “left”, where the similar phoneme in the word seems to make the prediction more difficult, other word pairs where a similar difficulty can be observed are “no” and “nine” as well as “on” and “one”. These issues along with the difference in validation and testing accuracies, show that simply using the raw waveform data as an input is not enough to create a perfect representation of the data, both in terms of generalizing to words outside the scope of the training set as well as dealing with noise, whether its environmental background audio or noise caused by poor quality recordings.

6 CONCLUSION

This work has shown that a model using one-dimensional convolutions along with sensible training techniques can effectively be used for single-word speech recognition, but more work needs to be done on generalizing to unseen samples and dealing with extremely noisy samples.

6.1 Discussion

This section discusses the pre-experiments on model architecture, data augmentation techniques and training setup. As the experiments were explorative in nature and the results achieved aren't conclusive, results from them haven't been reported in the thesis and thus this section will be more of an open discussion covering what worked and what didn't in the experiments.

The final model architecture is relatively simple. More complex model architectures that resemble ResNet (He et al. 2016) and DenseNet (Huang et al. 2017) did not achieve test results as good. Various kernel sizes for both the convolutional layers and max-pooling layers were tested. A small kernel size for the convolutional layers worked very well, the difference between the sizes 3, 5 and 9 was not significant, but the kernel size of 5 was chosen as it performed slightly better. The kernel size 4 for max-pooling layers was chosen as it provides a middle ground between the amount of pooling layer used as well as allowing the use of multiple consecutive convolutional layers between each pooling operation without the model getting too deep. Replacing max-pooling with convolutions using a larger stride didn't perform as well as when using max-pooling. Other architectural experiments included using global max-pooling instead of global-average pooling between the convolutional and fully-connected layers, using both global max-pooling and global average-pooling and simply flattening out the output from the convolutional layers. Simply using global average-pooling, as in the work by Dai et al. (2017) worked the best. Not only do the other two methods use more parameters, they also performed worse in both validation and testing.

Various types of data augmentation were used during testing, ranging from simply scaling the amplitude of the input to changing the pitch. The effect of each augmentation technique varied significantly. Mixing noise with the input showed to be the most effective

augmentation technique, this effect was slightly improved further by randomly scaling the amplitude of the input. Shifting the input in time also showed to be a useful technique. Stretching the input in time, changing the pitch and randomly altering the speed of the input basically showed no effect, or in some cases even a negative effect on the performance of the model.

Two methods for using pseudo-labels during training were tested. The first method was to keep training the model further, by doing a warm restart in the same way as during the original training process. The second method was to fully re-train the model. Completely re-training the model achieved better results in the end, but it also required more training time. If training-time would be a bottleneck, it would be recommended to use the method of continuing training after a warm restart.

The learning rate schedule was chosen based on the results by Loshchilov and Hutter (2016). Other SGRD learning rate schedules were experimented with and they achieved similar results. Other optimizers like Adam (Kingma, Ba 2014) converged faster but did not generalize as well to the test data.

6.2 Future work

While the current model is small compared to computer vision models like VGGNet (Simonyan, Zisserman 2014), it is relatively large compared to models intended for running on mobile devices and microcontrollers like the models used by Zhang et al. (2017). Many steps could be taken to decrease the model size, at varying degrees of cost in performance. A smaller kernel size could be used in the convolutional layers, alternatively the smaller kernel size could be combined with dilated convolutions (Yu, Koltun 2015) in order to maintain the same receptive field but the parameter count of the smaller kernel size. Apart from tuning hyperparameters, methods such as pruning convolutional kernels could be used. Molchanov et al. (2016) showed that pruning can be a very effective technique to speed up inference time, with only limited effects on overall performance.

Most real-life use cases require classification from a continuous audio stream. This requires some sort of posterior-handling which could be done by smoothing out frame-based posteriors and computing a confidence score for a smoothed-out window of frames. A final decision would be made if the confidence score exceeds a predefined threshold-value. (Chen, Parada & Heigold 2014)

While this work focused on only using a single model for the entire process, results could be quite easily improved with the use of ensembles. Ensembles of multiple models could be used both for generating pseudo-labels as well as the final predictions. Generating only the pseudo-labels with an ensemble of models would theoretically allow for higher quality pseudo-labels while keeping the parameter count low, as only one model would be used for inference.

REFERENCES

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G. and Chen, J. 2016, Deep speech 2: End-to-end speech recognition in english and mandarin. In International Conference on Machine Learning (pp. 173-182).

Badrinarayanan, V., Kendall, A. and Cipolla, R. 2017, Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12), pp. 2481-2495.

Chen, G., Parada, C. and Heigold, G. 2014, May. Small-footprint keyword spotting using deep neural networks. In Acoustics, speech and signal processing (icassp), 2014 ieee international conference on (pp. 4087-4091). IEEE.

Dai, W., Dai, C., Qu, S., Li, J. and Das, S, 2017, March. Very deep convolutional neural networks for raw waveforms. In Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on (pp. 421-425). IEEE.

Deng, J., Dong, W., Socher, R., Li, L.J., Li, K. and Fei-Fei, L. 2009, Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on (pp. 248-255). IEEE.

Glorot, X., Bordes, A. and Bengio, Y. 2011, June. Deep sparse rectifier neural networks. In Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (pp. 315-323).

Hannun, A.Y., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A. & Ng, A.Y. 2014, Deep Speech: Scaling up end-to-end speech recognition, CoRR, vol. abs/1412.5567.

He, K., Zhang, X., Ren, S. and Sun, J. 2016, Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, CoRR, vol. abs/1704.04861.

Huang, G., Liu, Z., Weinberger, K.Q. and van der Maaten, L. 2017, Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (Vol. 1, No. 2, p. 3).

Ioffe, S. and Szegedy, C. 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In International Conference on Machine Learning (pp. 448-456).

Kim, Y. 2014, Convolutional Neural Networks for Sentence Classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1746-1751).

Kingma, D.P. & Ba, J. 2014, Adam: A Method for Stochastic Optimization, CoRR, vol. abs/1412.6980.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. 2012, Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L. 2014, September. Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

LeCun, Y., Bengio, Y. and Hinton, G. 2015, Deep learning. nature, 521(7553), p. 436.

LeCun, Y., Boser, B.E., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W.E. and Jackel, L.D. 1990, Handwritten digit recognition with a back-propagation network. In Advances in neural information processing systems (pp. 396-404).

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. 1998, Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp. 2278-2324.

Lee, D.H. 2013, Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Workshop on Challenges in Representation Learning, ICML (Vol. 3, p. 2).

Loshchilov, I. & Hutter, F. 2016, SGDR: Stochastic Gradient Descent with Restarts, CoRR, vol. abs/1608.03983.

McKinney, W. 2010, Data structures for statistical computing in python. In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51-56).

Molchanov, P., Tyree, S., Karras, T., Aila, T. & Kautz, J. 2016, Pruning Convolutional Neural Networks for Resource Efficient Transfer Learning, CoRR, vol. abs/1611.06440.

Oliphant, T.E. 2006, A guide to NumPy (Vol. 1, p. 85). USA: Trelgol Publishing.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L. and Lerer, A. 2017, Automatic differentiation in PyTorch.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V. and Vanderplas, J. 2011, Scikit-learn: Machine learning in Python. Journal of machine learning research, 12(Oct), pp.2825-2830.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. 2016, You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

Sainath, T.N. and Parada, C. 2015, Convolutional neural networks for small-footprint keyword spotting. In Sixteenth Annual Conference of the International Speech Communication Association.

Schmidhuber, J. 2015, Deep learning in neural networks: An overview. Neural networks, 61, pp.85-117.

Simonyan, K. & Zisserman, A. 2014, Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR, vol. abs/1409.1556.

Smith, N. 2014, No more pesky learning rate guessing games, CoRR, vol. abs/1506.01186.

Springenberg, J., Dosovitskiy, A., Brox, T. and Riedmiller, M. 2015, Striving for Simplicity: The All Convolutional Net. In ICLR (workshop track).

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R. 2014, Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), pp.1929-1958.

Sukhbaatar, S. & Fergus, R. 2014, Learning from Noisy Labels with Deep Neural Networks, CoRR, vol. abs/1406.2080.

Tieleman, T. and Hinton, G. 2012, Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning, 4(2), pp.26-31.

Warden, P. 2017a, Launching the speech commands dataset. Google Research Blog.

Warden, P. 2017b, Speech Commands: A public dataset for single-word speech recognition, Dataset available from http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. and Klingner, J. 2016, Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR abs/1609.08144.

Zhang, Y., Suda, N., Lai, L. & Chandra, V. 2017, Hello Edge: Keyword Spotting on Microcontrollers, CoRR, vol. abs/1711.07128.

APPENDIX 1. SWEDISH SUMMARY

Taligenkänning av enskilda ord med Konvolutionella neuronnät på rå ljuddata

Djupinlärning (*eng. deep learning*) och neuronnät har uppnått toppresultat inom flera olika fält inom maskininlärning som t.ex. bildklassificering (Simonyan, Zisserman 2014), maskinöversättning (Wu et al. 2016) och röstigenkänning (Amodei et al. 2016). Dessa neuronnät kan bestå av tiotals miljoner parametrar, där både träning och inferens kräver mycket tid samt kraftfull hårdvara. Medan dessa modeller uppnår toppmoderna resultat, kan de inte användas effektivt på mindre kraftfulla enheter som mobiltelefoner samt inbäddade system. Modell arkitekturer som MobileNet av Howard et al. (2017) har skapats för mer effektiv inferens samt lägre minnesanvändning där man fortfarande uppnår jämförbara resultat gentemot den absoluta toppen.

Det här examensarbetet fokuserar av taligenkänning på enskilda ord eller nyckelords igenkänning (*eng. keyword spotting, KWS*), där målet är att känna igen en uppsättning av fördefinierade ord från korta ljudklipp. Zhang et al. (2017) diskuterar behovet av att ha modeller med hög noggrannhet samt låga beräknings krav för att effektivt användas på enheter som mobiltelefoner och inbäddade system. De diskuterar problem med alltid-på, över nätverket, röstigenkänning på enheter som t.ex. Amazon Echo och Google Home, där det inte endast är ineffektivt att kontinuerligt strömma ljud över nätverket, men det leder även till problem med integritetsfrågor. De nämner även hur en del av problemen lindras genom användning av KWS, där enheten endast lyssnar för enstaka nyckel-ord eller fraser som “OK Google”. Efter igenkänningen aktiveras det fulla röstigenkänningssystemet. Detta tillåter enheten att använda mer komplexa modeller endast när det behövs och lättar på en del av integritetsfrågorna.

Fokus med arbetet är användningen av ett konvolutionellt neuronnät (*eng. Convolutional Neural Network, CNN*) som använder sig av endimensionella konvolutioner på rå ljuddata. Såväl som effektiva inlärningsmetoder för att skapa en modell som uppnår högsta möjliga noggrannhet. Detta arbete gjordes som en del av TensorFlow Speech Recognition

Challenge[4] på Kaggle, en plattform för datavetenskap- samt maskininlärningstävlingar. Målet med tävlingen var att uppnå högsta noggrannhet på testdatamängden försedd av Kaggle, där etiketten för datat inte gavs ut och resultaten är baserade på inlämningen av en förutsägelsefil via Kaggle nätsidan.

Medan huvudanvändningsfallen för modeller som används i detta arbete är smarta enheter och inbäddade system, så fokuseras inte på detta specifika ändamål. Men användningsfallen beaktas både i design av modellarkitekturen såväl som resultaten. I verkliga användningsfall krävs ofta klassificering inom kontinuerliga ljudströmmar. Arbetet tar inte hänsyn till detta problem då syftet är på att klassificera testdatamängden som enbart kräver en förutsägelse per prov.

Arbetet använder sig av datamängden Speech Commands (Warden 2017) som består av ca 65 000 en sekund långa uttalanden om 30 ord som t.ex. ”yes”, ”no”, ”right” och ”left”. Målet är att lära sig klassificera tio av dessa ord samt två separata klasser där den ena representerar alla andra ord (”unknown”) och den andra representerar inget ord dvs. tystnad (”silence”). Testdatamängden, även försedd av Kaggle består av ytterligare 150 000 uttalanden. Dessa innehåller både ord och uttalare som inte finns i tränings datamängden.

För att nå den slutliga modellarkitekturen samt de använda träningsmetoderna genomfördes ett flertal experiment där olika arkitekturer testades i samband med olika träningstekniker och hyperparametrar.

Den slutliga modellarkitekturen är starkt inspirerat av VGGNet (Simonyan, Zisserman 2014) samt modellerna av Dai et al. (2017). Modellen består av sex block där varje block består av två konvolutionslager, satsnormaliseringslager (Ioffe, Szegedy 2015) samt Rectified Linear Unit (ReLU) aktiveringsfunktioner. Varje block följs av ett pooling-lager där alla förutom sista lagret använder sig av max-pooling, det sista lagret använder global average pooling. Konvolutionslagren använder sig av endimensionella konvolutioner med en filterstorlek av 5, max-pooling lagren har en filterstorklek av 4. Det första blockets konvolutionslager har 8 filter, följt av en fördubbling av filter för varje block därefter. Modellens klassificeringsdel består av två lager med 128 och 64 neuroner, de följs av ReLU aktiveringsfunktioner och dropout-lager med en utfallshastighet av 0.5.

[4] https://www.kaggle.com/c/tensorflow-speech-recognition-challenge

Det sista lagret består av 12 neuroner för att motsvara antalet klasser i datamängden. Den totala mängden parametrar i modellen är ca. 0.7 millioner.

För effektiv träning av modellen används fyra huvudsakliga tekniker: viktad slumpmässig provtagning, Stochastic Gradient Descent with Warm Restarts (SGDR), användning av pseudo-etiketter samt artificiell utvidgning an träningsdatamängden (*eng. data augmentation*). Data augmentering (*eng. data augmentation*) är ett sätt att öka mängden träningsdata genom att modifiera den tillgängliga datan på ett sätt där det fortfarande kan identifieras som samma klass. Viktad slumpmässig provtagning hjälper med klassbalansen vid träning med att ge varje dataexemplar en vikt beroende på den totala mängden exemplar av den klassen. Vikten används sedan för att slumpmässigt välja exemplar för varje tränings iteration. SGDR är en träningsmetod föreslagen av Loshchilov och Hutter (2016) där modellens parametrar optimeras med en gradvis sjunkande inlärningstakt (*eng. learning rate*) med ett antal omstarter där inlärningstakten återställs till sitt ursprungliga värde och sänkningsprocessen omstartas. Användningen av pseudo-etiketter består av att först med en tränad modell göra förutsägelses på omärkt data, i detta fall testdatamängden, och använda de mest självsäkra förutsägelserna som sanna etiketter (Lee 2013).

Modellen uppnår 97,4% noggrannhet på valideringsdatamängden, samt 88,7% och 89,4% noggrannhet på de två testdatamängderna som består av 30% respektive 70% av den totala testdatamängden. Den höga valideringsnoggrannheten tyder på att modellen effektivt kan lära sig representationer för de ord som det ser under träningsprocessen. Granskning av förutsägelser visar att testdatamängden är mycket svårare än valideringsdatamängden, mängden oljud och bakgrundsljud är mycket högre, samt är en del av orden avkapade så att man inte kan säga vilket or det är frågan om. Modellen har också svårigheter med ord som är utanför träningsdatamängdens omfattning som t.ex. ”backward” or ”learn”. Ordet ”learn” är ofta felaktigt förutsedd som order ”left”, vilket tyder på svårigheter med ord som innehåller liknande fonem. Andra ordpar med liknande svårigheter är “no” och “nine” samt “on” och “one”. Dessa problem visar att användningen av endast rå ljuddata samt CNN är inte tillräckligt för att perfekt generalisera till nya ord samt exemplar med mycket oljud eller bakgrundsljud.

KÄLLOR

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G. and Chen, J. 2016, Deep speech 2: End-to-end speech recognition in english and mandarin. I International Conference on Machine Learning (s. 173-182).

Dai, W., Dai, C., Qu, S., Li, J. and Das, S. 2017, Very deep convolutional neural networks for raw waveforms. I Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference (s. 421-425). IEEE.

Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M. & Adam, H. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, CoRR, vol. abs/1704.04861.

Ioffe, S. and Szegedy, C. 2015, Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In International Conference on Machine Learning (s. 448-456).

Lee, D.H. 2013, Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. I Workshop on Challenges in Representation Learning, ICML (Vol. 3, s. 2).

Loshchilov, I. & Hutter, F. 2016, SGDR: Stochastic Gradient Descent with Restarts, CoRR, vol. abs/1608.03983.

Simonyan, K. & Zisserman, A. 2014, Very Deep Convolutional Networks for Large-Scale Image Recognition, CoRR, vol. abs/1409.1556.

Warden, P. 2017, Speech Commands: A public dataset for single-word speech recognition, Tillgänglig: http://download.tensorflow.org/data/speech_commands_v0.01.tar.gz

Wu, Y., Schuster, M., Chen, Z., Le, Q.V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K. and Klingner, J. 2016, Google's neural machine translation system: Bridging the gap between human and machine translation. CoRR abs/1609.08144.

Zhang, Y., Suda, N., Lai, L. & Chandra, V. 2017, Hello Edge: Keyword Spotting on Microcontrollers, CoRR, vol. abs/1711.07128.