CIFAR-10 Image Recognition

EE4305 Introduction to Fuzzy/Neural Systems

Mario Gini
Thomas Michael Hayden

ETH Zurich & University of Oxford

Contents

1 Introduction

T. Hayden & M. Gini

The CIFAR-10 dataset contains 60000 images bla bla.

Objectives of this project are: bla bla
Structure of the report is as follows: Section 2 gives a general literature review.

2 Literature Review on Artificial Neural Networks

M. Gini

This section gives a literature review on the broad topic of artificial neural networks (ANN). A more specific review on ANN designed to classify the CIFAR-10 dataset is found in Section 3. The significance and applications of ANN will be reviewed in Section 2.1 while recent trends and accomplishments are discussed in Section 2.2.

2.1 Significance and Applications of Artificial Neural Networks

This subsection will illustrate the significance and applications of ANN. Increasing computer power shifted the focus of research towards deep ANN and similar architectures which are coined under the term "deep learning". These powerful deep ANN are nowadays used in a variety of applications [1] [2].

ANN are significant because they can work as a black box model. The performance can be improved by data preprocessing, augmentation and mainly by finding an appropriate network architecture and training process. No a-priori knowledge of the classification process itself is required. This makes deep ANN suited for applications where such knowledge is difficult to obtain. Character and speech recognition are such difficult problems, as well as image classification. In speech recognition, deep ANN have been shown to outperform other methods on a variety of speech recognition benchmarks, sometimes by a large margin [3]. In the field of image classification, the 2012 ILSVRC (ImageNet Large-Scale Visual Recognition Challenge) marks an important turning point because a convolutional neural network (CNN) architecture won the competition for the first time - by a large margin [4]. In both fields, ANN are now widely accepted as the most powerful approach.

However, the fact that ANN do not incorporate much a-priori knowledge can also backfire. In consequence, a trained model gives little insight into its inner workings and optimal network architectures are basically found through a trial-and-error process. Most design guidelines for deep learning methods are therefore rather based on empirical knowledge than on theoretical foundations.

New methods are developed to better understand the computations deep ANN perform at each layer. The resulting visualizations reveal the process of extracting high level features out of raw input data [5] [6]. In general, each layer extracts higher level features of the input the previous layer provides such that the features are highly abstract after a few layers. The last layer then classifies the input into one of the output categories.

2.2 Recent Trends and Accomplishments

Recent trends and accomplishments of ANN are described in this subsection. Two recent accomplishments are looked at in detail: The AlphaGo computer program and adversial examples. AlphaGo is a great example to illustrate the great capabilities of ANN. Adversial examples can easily fool very different kinds of neural networks which is a good way to exemplify the limitations the present ANN still possess.

The game Go is a complex board game with the impressive number of around 10^{170} legal positions [7]. Due to its enormous search space and difficulty to evaluate board positions, it is viewed as the most challenging of the classical games for artificial intelligence. A victory of a computer program over a professional human player has been considered to be at least a decade away. However, the computer program AlphaGo beat the European Go champion 5-0 in 2015 [8].

AlphaGo makes extensive use of ANN. It consists of a "value" and a "policy" network to separately evaluate the board position and select moves. It is trained in a combination of supervised learning from human expert games and reinforcement learning through self-play. The training of such big networks requires notable computation resources. In a recent trend, dedicated hardware to train deep ANN is developed. Besides other adaptions, it is designed to speed up matrix multiplications which are one of the main components of the training process. The most notable example is the Tensor Processing Unit which achieves a 15- to 30-fold performance compared to a contemporary GPU or CPU [9]. It is important to note that the development of deep learning is closely connected to the ever improving available computing power [10].

AlphaGo received considerable media coverage and is considered as one of the most impressive feats of deep learning. In a follow-up paper, a further improved version of AlphaGo is presented, AlphaZero[11]. It uses a single neural network and trains solely through reinforcement learning with self-play, starting with random play. It is only provided with the rules of Go. After only days of training, it defeated all previous versions of AlphaGo and achieved a never seen before playing strength. It is quite intriguing that even for such a complex task, the network can achieve superhuman performance without any provided knowledge besides the rules of the game.

As a second recent trend, adversarial examples recently surprised a lot of researches and became a hot topic of interest. To generate an adversarial example, a slight perturbation is applied to a correctly classified image. The classification process is then repeated and the perturbation is adapted such that the prediction error is *maximized*. A slight perturbation which is not recognizable by a human is already enough to let the neural network misclassify an image with a high confidence level[12]. It has been shown that adversarial examples trained on one model are likely to be misclassified by another model as well, i.e. they possess a transferability property[13].

It is very likely that a randomly selected input to a neural network built from linear parts is processed incorrectly and the models only behave reasonably on a very thin manifold encompassing the training data[14]. This result questions the generalization abilities of ANN. Furthermore, the transferability property allows potential attacks on systems using ANN[15][16]. For example, stop signs could be slightly modified with stickers such that they are misclassified by autonomous vehicles which then behave unexpectedly. Further research is required to develop defense strategies against such attacks. Only then, ANN can deployed in safety critical applications.

3 Literature Review on the CIFAR-10 dataset

T. M. Hayden

The CIFAR-10 dataset[17] is a well established dataset in the machine learning community. It is challenging because it is a relatively small dataset. Even so, excellent results, even exceeding human performance, have been obtained using a variety of CNN architectures[1]. At the time of writing, the highest published result on the CIFAR-10 dataset was achieved in 2015 with accuracy of 96.53%. This is considerably better than human performance which has an accuracy of around 94%[18].

3.1 Data Augmentation

Like many other machine learning problems, image classification will almost always benefit from additional data[19]. However, even when restricted to a particular dataset such as CIFAR-10 it is possible to generate more data using a technique called data augmentation[20]. Data augmentation manipulates existing images to create 'new' data for use in training.

Common methods to augment images for use in machine learning include mirroring, rotation and image translation[4]. Using these techniques it is possible to train on a dataset that can be several times larger than the original dataset. The leading architectures all made heavy use of data augmentation[21][22][23].

[1] `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html`

3.2 State of the art architectures for classifying the CIFAR-10 dataset

In this section, the results of several different CNN architectures are presented. It should be noted that these architectures were not designed specifically to perform on the CIFAR-10 dataset. As such, they may not be fully optimised and it is likely that they could be improved slightly.

3.2.1 Fractional max-pooling

In a standard CNN, convolutional layers are often interspaced with 2x2 max-pooling layers. These max-pooling layers serve to downsample the data. This allows the CNN to be somewhat spatially invariant to the locations of the features and improve accuracy. However each max pooling layer also removes 75% of the data[21]. This in effect reduces the maximum depth of the CNN due to the disjoint nature of the max-pooling regions.

By using a new approach known as fractional max-pooling, it is possible to max-pool using a non-integer mask size. In this manner, the size of the hidden layers is reduced by a lesser amount and it is possible to create deeper networks without having to add consecutive convolutional layers. This is important as generally deeper networks will lead to stronger classifiers[24]. However, deeper networks are also in general more expensive to train.

An architecture based on fractional max-pooling currently has the highest published classification accuracy on the CIFAR-10 dataset at 96.53%. This architecture also made heavy use of data augmentation. Additionally, the model was 'fine-tuned' after initial training by re-training on the original dataset for a few epochs using a low learning.

3.2.2 The All Convolutional Net(ALL-CNN)

In this architecture[23] a CNN consisting entirely of convolutional layers is proposed. Max-pooling layers are instead replaced with convolutional layers with increased stride. These increased stride layers act in a similar way to max-pool layers in that they downsample the data and allow the CNN somewhat invariant feature location. This architecture has an accuracy of 95.59% which is the 2nd highest published result. This architecture also makes heavy use of data augmentation.

3.2.3 Layer-sequential unit-variance (LSUV) initialization

LSUV initialisation provides a method to initialise deep CNN. This produces networks with better accuracy than uninitialised networks. In addition LSUV greatly accelerates the training of CNNs. An architecture based on the LSUV method machine managed to achieve an accuracy of 94.16%. Note that this only used a moderate amount of data augmentation.

It is important to stress the use of data augmentation when looking at these results. Table shows the results of the three leading architectures along with the amount of data augmentation. Moderate data augmentation consists of mirroring in the horizontal axis and small translations in each axis. Extreme data augmentation involves upscaling the images to 126×126 pixel images and performing a variety of operations such as shearing, colour augmentation, rotation, translation and scaling. It may be the case that with additional data augmentation, LSUV outperforms the max-Pooling approach.

3.3 Application areas of image recognition algorithms

There are numerous applications of image recognition algorithms across many different fields. However in order for neural networks to be effective, large amounts of labeled data must first be collected. In practice labelling and uploading of images is mostly done by users of the

Data Augmentation	Fractional Max-Pooling	ALL-CNN	LSUV
None	-	90.92%	-
Moderate	-	92.75%	93.94%
Extreme	96.53%	95.59%	-

Table 1: Table showing the results of the leading CIFAR-10 architectures.

application by 'tagging' images. For example in a stock image database such as Shutterstock[2], users would be prompted to tag uploaded images with the image contents. This provides Shutterstock with an enormous amount of data perfect for machine learning. This has allowed Shutterstock to develop powerful new tools to label images using machine learning. For example the newly released compositionally aware search which allows users to serarch for images with specific objects in different locations of the image[25].

Another application of image recognition is to automatically tag recognised faces when uploading photos onto social media websites. This is useful as it is often tedious to tag each image in large albums. Automatic tagging algorithms have been developed using machine learning to automatically tag faces with accuracy as high 99%[26]. This allows users to upload entire albums without having to tag each photo individually.

4 Multi-Layer Perceptron Classifier

M. Gini & T. M. Hayden

This section presents the multi-layer perceptron (MLP) classifier designed to classify the CIFAR-10 dataset. It is organized as follows: Section 4.1 introduces the software setup used to implement the MLP classifier. Section 4.2 discusses the data preprocessing and augmentation. Section 4.3 analyzes the effect of different network structures on performance. Section 4.4 analyzes the effect of different hyperparameters.

4.1 Software Setup

MATLAB's Neural Networks toolbox is employed to implement the MLP classifier. The toolbox provides convenient algorithms and applications to design the MLP. A network training function with a convenient graphical user interface (GUI) to observe the progress is included as well. Figure 1 shows the GUI.

Since this is a classification problem, parts of the network structure are given. The output of the MLP should be a prediction of to which class the input belongs to. This is accomplished with the help of the softmax function, also called normalized exponential function. Equation 4.1 shows the formula of such a function. A softmax layer is then used as the last layer. It gets a K-dimensional input vector $\boldsymbol{z}$ of arbitrary real values and "squashes" it into a K-dimensional output vector $\sigma(\boldsymbol{z})$ of real values in the range $[0, 1]$. In our case, $K = 10$ and the output values represent the probabilities that the input belongs to the respective class. The class with the highest probability then constitutes the predicition of the MLP.

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \quad \text{for} \quad j = 1, \ldots K. \tag{4.1}$$

[2] https://www.shutterstock.com/

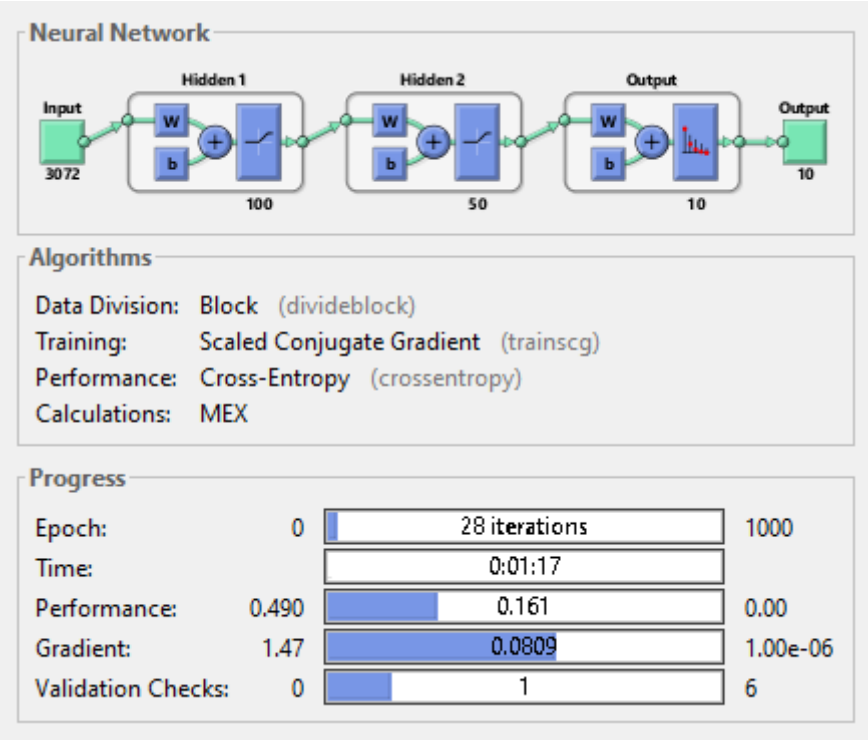

Figure 1: Screenshot of MATLAB's nntraintool.

The other hidden layers consist of standard MLP. The input to the MLP are the pixel values of the image. Preprocessing and augmentation as described in the next section is also done before the pixel values are fed into the network.

MATLAB offers lots of adjustable settings for the MLP. Unless mentioned otherwise, the following settings are employed as default settings:

- Training function:
- Loss function: cross entropy
- Activation function: 'tansig', the hyperbolic tangent sigmoid activation function
- Training batch size: 20000 images
- Network structure: Two hidden layers, 100 neurons on the first layer and 50 neurons on the second layer.
- Dataset structure: 80% are used for the training and 20% are used for validation. The last 10000 images are specifically used for testing the performance of the MLP.

4.2 Data Preprocessing and Augmentation

Data preprocessing and augmentation take place before the data is fed into the network. In the preprocessing step, the data is normalized and centered around the mean. In the augmentation step, the amount of data is augmented through simple operations like image flipping.

4.2.1 Data Preprocessing

Each image of the dataset is represented by a 32*32*3 array, which results in 1024 pixel values per color channel. To be processed by the MLP, it is transformed into a 1*3072 array. The

pixel values are integers in the range [0,255]. For normalization, the data is divided by 255 to lie within the range [0,1]. Accordingly, the datatype is changed from the integer type to double. In a second step, the mean per pixel over the whole training set is subtracted. This centers the data per channel.

Data preprocessing also includes the division of the complete dataset into appropriate training, validation and test data batches. The CIFAR-10 dataset consists of 60000 images, with 10000 specifically labeled for testing. For our performance analysis, we always use the provided test batch. The effect of varying training batch size can be seen below here some plot of varying training size

4.2.2 Data Augmentation

Experience shows that a larger training data set increases network performance. A basic and still successful data augmentation method is vertical mirroring. here some plot of varying training size with mirrored data When comparing with above, the increase in performance can clearly be seen.

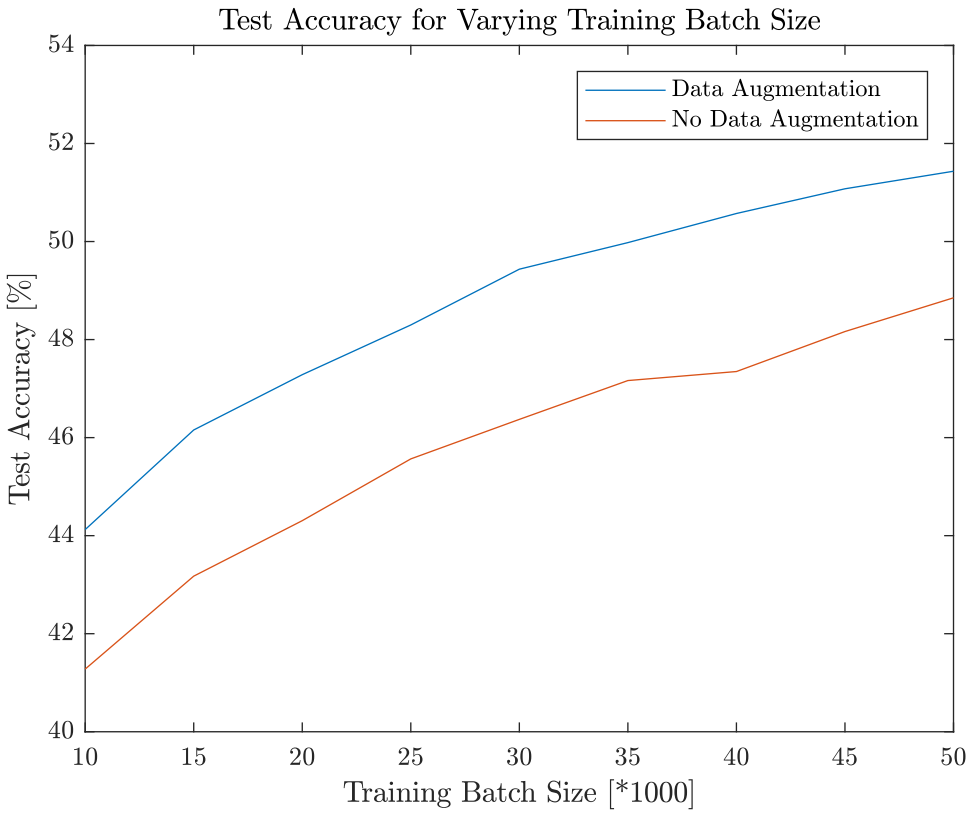

Figure 2: Hello Boy

4.3 Optimization of Network Structure

c)on the training of the MLP

- Varying the number of neurons
- Varying the number of layers

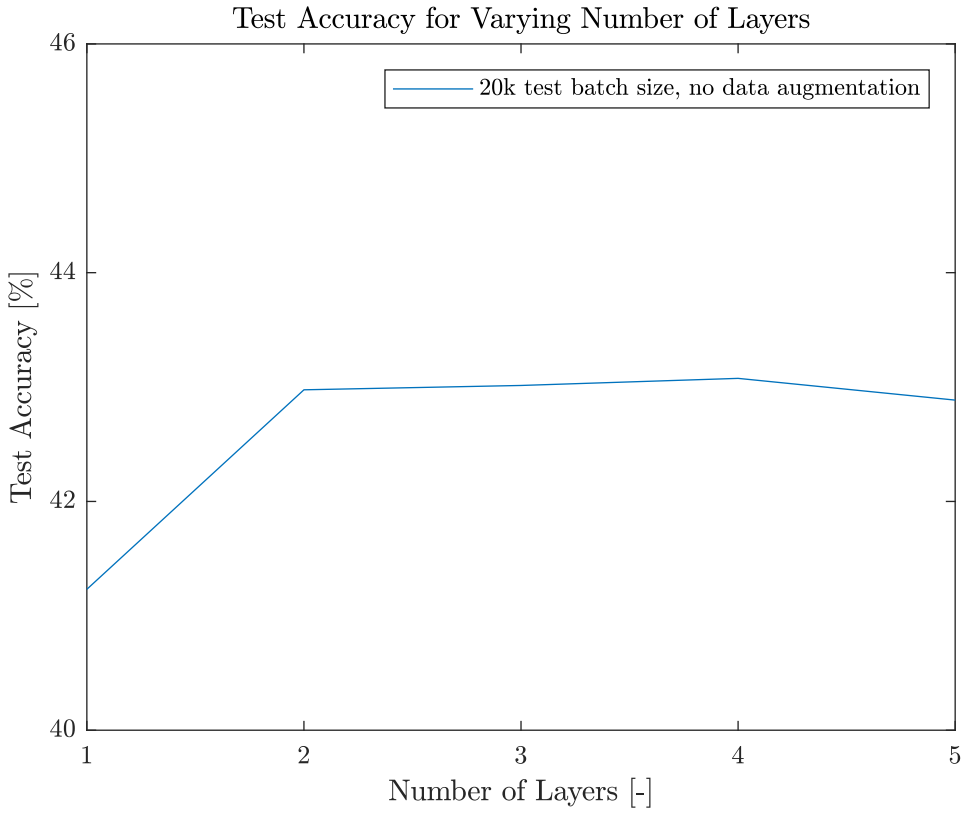

Figure 3: Hello Boy2

4.4 Optimization of Network Hyperparameters

d) on the performance of the MLP with different objective functions and optimization methods
Some pic of the confusion matrix

- Different learning rates
- Different optimization methods
- Different performance functions There are six different performance functions available in the MATLAB environment:

e)any other interesting observation that you think are pertinent (e.g. effect of learning rate on convergence speed).

5 CNN network

M. Gini & T. M. Hayden

6 Conclusion

M. Gini & T. Hayden

Long story short: we completely aced our project BOOM
2+2 = 4 -1 = 3 quick maths

Bibliography

[1] Li Deng, Dong Yu, et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, 7(3–4):197–387, 2014.

[2] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[3] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[4] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[5] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going Deeper into Neural Networks, June 2015. URL `http://googleresearch.blogspot.com/2015/06/inceptionism-going-deeper-into-neural.html`.

[6] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.

[7] John Tromp and Gunnar Farnebäck. Combinatorics of go. In *International Conference on Computers and Games*, pages 84–99. Springer, 2006.

[8] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[9] Norman P Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, et al. In-datacenter performance analysis of a tensor processing unit. *arXiv preprint arXiv:1704.04760*, 2017.

[10] Jim X Chen. The evolution of computing: Alphago. *Computing in Science & Engineering*, 18(4):4–7, 2016.

[11] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[12] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[13] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[15] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[16] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, pages 506–519. ACM, 2017.

[17] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[18] Andrej Karpathy. Lessons learned from manually classifying cifar-10. *Published online at http://karpathy. github. io/2011/04/27/manually-classifying-cifar10*, 2011.

[19] Alon Halevy, Peter Norvig, and Fernando Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.

[20] Xiaodong Cui, Vaibhava Goel, and Brian Kingsbury. Data augmentation for deep neural network acoustic modeling. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(9):1469–1477, 2015.

[21] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.

[22] Dmytro Mishkin and Jiri Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[23] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Mike Ranzinger, Nicholas Lineback, and Nathan Hurst. Composition aware search.

[26] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.