Implementation and Optimization of Convolutional Neural Networks for Plankton Classification

Darin Tsui*, Aniket Dhar*
*Shu Chien - Gene Lay Dept. of Bioengineering
University of California San Diego, La Jolla, CA 92093

***Abstract*—With rapidly increasing climate change, comprehensive metrics of ocean health are vital to understanding how human actions are altering the populations of various species within the ocean. One common metric of ocean health is the population of various species of plankton within the ocean, as they form the basis of many species' meals and are notably susceptible to man-made pollution, particularly harmful chemicals [1]. In order to process the large amounts of image data collected by underwater imaging sensors, we evaluated the usage of convolutional neural networks to perform image classification. Our team preprocessed the image data and trained CNN models on the data, and were able to optimize the hyperparameters to achieve a maximum test accuracy of 74% using AlexNet.**

***Index Terms*—Machine Learning, CNN, Image Classification, AlexNet**

I. Introduction

Convolutional neural networks (CNNs) have become one of the most popular and effective deep learning architectures for computer vision applications [2]. The main premise behind CNNs is their ability to recognize patterns in images or videos through the use of convolutional layers [3]. These convolutional layers serve to extract features from input data and have been shown to be highly effective in a wide range of applications [4].

While the term "CNN" can imply a wide variety of models, some of the most popular CNNs include models such as AlexNet, VGGNet, and ResNet [5]. Part of their popularity results from achieving state-of-the-art performance on benchmark datasets, as well as heavily optimizing the number of convolutional layers used.

While CNNs have been shown to be effective in many applications, CNN architecture design has revolutionized the field of computer vision for its high performance in applications that require image recognition, object detection, and image segmentation [6]. The main reason for the rise of CNNs in computer vision is their ability to earn and extract features from extremely large datasets. This makes the CNN architecture ideal for solving highly complex problems. However, it is not well known how tuning specific parameters can affect CNN performance. In this paper, we perform a deep dive into CNN architecture using the 2014 National Data Science Bowl competition dataset.

II. Methods

In this section, we describe a detailed description of the architectures used. Then, we describe the dataset used, as well as the preprocessing steps used in our approach.

A. Baseline CNN

CNNs have long been among the favored algorithms for image classification due to their ability to downsample large image files without losing data through their convolutional layers. They have also not been noted to experience extreme overfitting when trained on a large corpus of data, a common pitfall in other algorithms. As such, we chose to leverage a simple CNN architecture comprised of 2 convolutional blocks and a linear layer. The sample architecture is shown in Figure 1. Cross-Entropy loss was then applied to the output of the final layer to yield a class output.

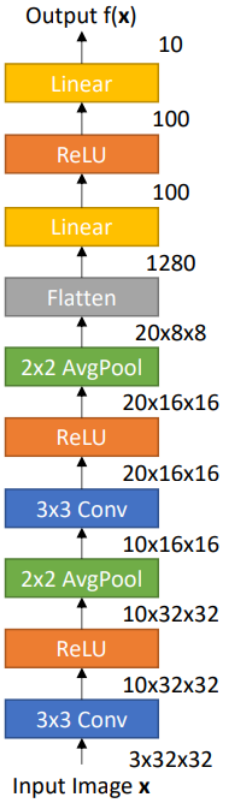

Figure 1: Architecture of Baseline Model

B. AlexNet

AlexNet is a CNN architecture purposed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012 [7]. The idea that AlexNet popularized was the usage of Rectified Linear Units (ReLU) as its activation function [8]. Equation

1 gives the equation for ReLu. The purpose of ReLu is to constrain the activation function to a value greater than or equal to 0, and thus introduce nonlinearity and is non-saturating [9]. In practice, it was found that ReLu improves generalization performance and reduces training time.

$$f(x) = max(0, x) \tag{1}$$

The CNN architecture of AlexNet consists of using 8 layers, with 5 of those layers being convolutional layers and 3 of those layers being fully connected layers [8]. As its input, AlexNet takes in an RGB image of 227 x 227 pixels. The architecture also adds dropout layers to the model. Figure 2 displays a visual representation of the AlexNet architecture. The network is traditionally split into two copies in order to leverage the computing power of Graphics Processing Units (GPUs), excluding the last layer.

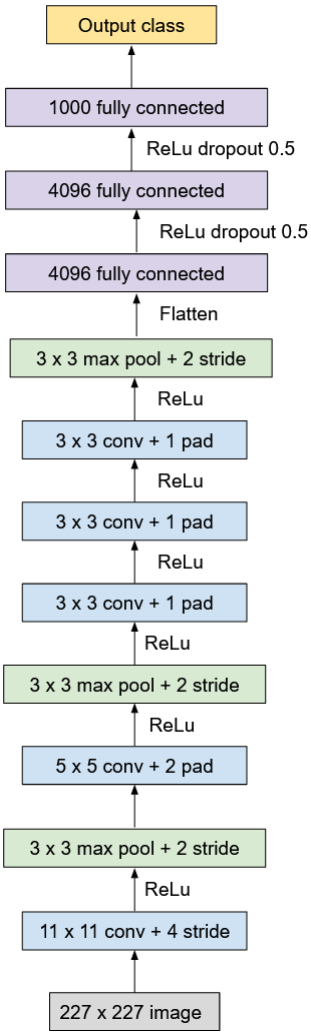

Figure 2: Architecture of AlexNet

C. Description of Data

The dataset used in this study was obtained from the National Data Science Bowl competition. In this dataset, Oregon State University's Hatfield Marine Science Center provided a large-scale dataset of 33k labeled gray-scaled images of plankton, coming from 121 different classes. Each raw image was automatically cropped to extract regions of interest, resulting in images of various sizes. These sizes can range from 50 x 30 pixels to 150 x 150 pixels [10]. Figure 3 displays the various classes present in the dataset, as well as their relative biological connections to each other.

In total, approximately 24k images were manually labeled for purposes of training and testing the CNN architectures. Upon visual inspection of the data, it is apparent that the dataset given is severely imbalanced. In particular, while some classes can have as many as 900 images, some classes can have as few as 30 images.

D. Preprocessing

To ensure that CNN architecture is able to be implemented, we resized all images such that they were 48 by 48 pixels. Additionally, the Red, Green, and Blue (RGB) values of the images were normalized in accordance with their relative weight in grayscale. For instance, the red value of each image was normalized such that its mean was 0.485. The green value was normalized to a mean of 0.456, and the blue value was normalized to a mean of 0.406.

The resized images were then saved to different folders based on their class. The total number of classes present in the dataset was similarly extracted from this list of classes. As the test dataset did not contain any classes by which to identify what each image represented, these images were simply saved to a separate folder to keep them separated from the labeled data.

Although the Hatfield Marine Science Center did provide a set of images to be used as a test sample, these images were not labeled with the class names [10]. As such, the labeled data was used for both training and testing in order to compare various models to one another based on accuracy.

III. Experiments

In this section, we provide a detailed description of the experiments done. Then, we discuss the performance of our CNN architectures under different setups.

Table I displays the default experimental hyper-parameters used in all setups unless otherwise specified. The dataset was split up into 80% training and 20% testing sets.

Table I: Default experimental hyper-parameters.

Hyper-parameter	**Value**
Batch Size	32
Number of Epochs	10
Learning Rate	0.01
Optimizer	SGD
Loss-function	Cross-entropy

A. Visual Inspection of Samples

Before hyperparameter tuning AlexNet, we sought to test our dataset on a CNN architecture that we've previously seen. As such, we adapt the Baseline architecture, which

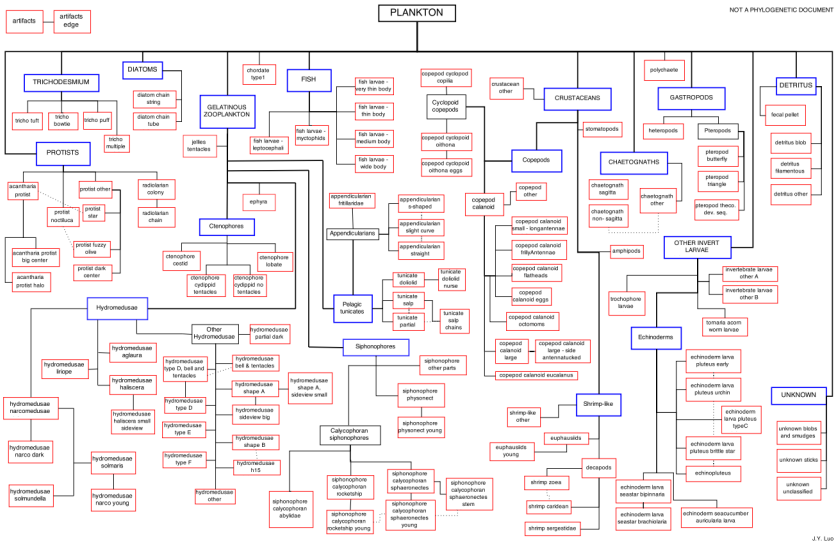

Figure 3: Visual representation of the classes present in the National Data Science Bowl competition. In total, 121 classes are present [10].

was presented in class, and plot a confusion matrix on the testing samples. Figure 4 displays the testing confusion matrix generated, based on the architecture from Figure 1.

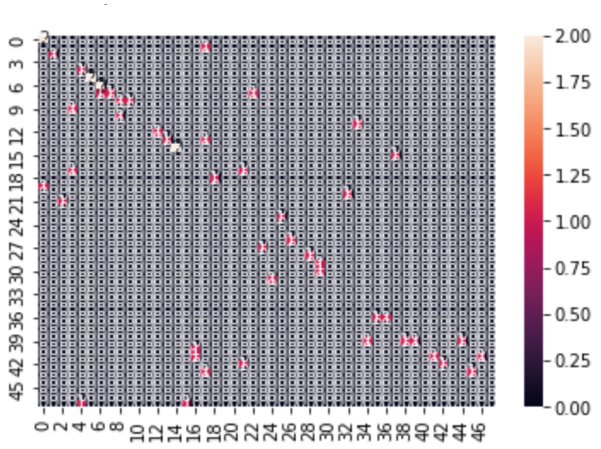

Figure 4: Visual inspection of samples via confusion matrix.

Based on Figure 4, we see that image classification tends to strongly misclassify a couple of specific images rather than throughout the matrix. Due to the large amount of data, these images were not able to be manually inspected. However, these results likely imply that certain plankton species appear visually similar to one another. This also could be the result of the imbalanced sampling of images, as certain classes of images may not show up frequently enough for the model to be able to properly differentiate them from other similar-looking species.

B. Hyper-parameter Tuning of Learning Rate

After exploring the effects of the baseline CNN on the dataset, we now move to understand the effects parameters can have on the final classification. To keep the results of our parameter tuning constant, we use AlexNet for this experiment as well as the rest of our experiments.

In the hyper-parameter tuning of learning rate, we chose to experiment with three learning rates: 1e-4, 0.001, and 0.01. Table II displays the final test accuracies recorded with each of the learning rates.

Table II: Test accuracies from varying learning rate.

Learning Rate	Test Accuracy
1e-4	9%
0.001	47%
0.01	60%

Additionally, Figure 5 displays the test accuracy and loss of learning rate with respect to epoch. From the diagram, we note that the learning rate of 0.01 was able to converge at around 65% accuracy, while the learning rate of 0.001 does not converge in the 10 epochs that the model was trained on. Interestingly, we note that in the majority of our runs, the model using a learning rate of 0.0001 tends to plateau far below the 65% benchmark, likely implying that the learning rate is too small to properly escape local minima, and thus is reported in this work. Since the learning rate of 0.01 appears to reach a consistent minimum of 65%, it appears to be the optimal learning rate of the models that we trained. We had also trained a learning rate of 0.1 model. However, this model did not show any signs of convergence and so was terminated.

C. Hyper-parameter Tuning of Optimizer

Here, rather than using stochastic gradient descent (SGD), we experiment with using the Adam optimizer **??**. The Adam optimization algorithm is an extension of SGD, and has seen widespread usage in various deep-learning applications (source).

Table III displays the final test accuracies using Adam.

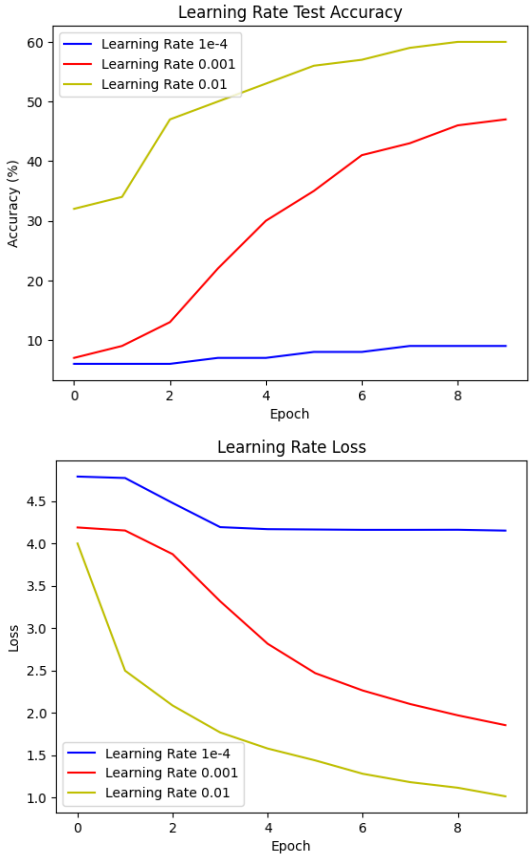

Figure 5: Learning rate test accuracy and loss.

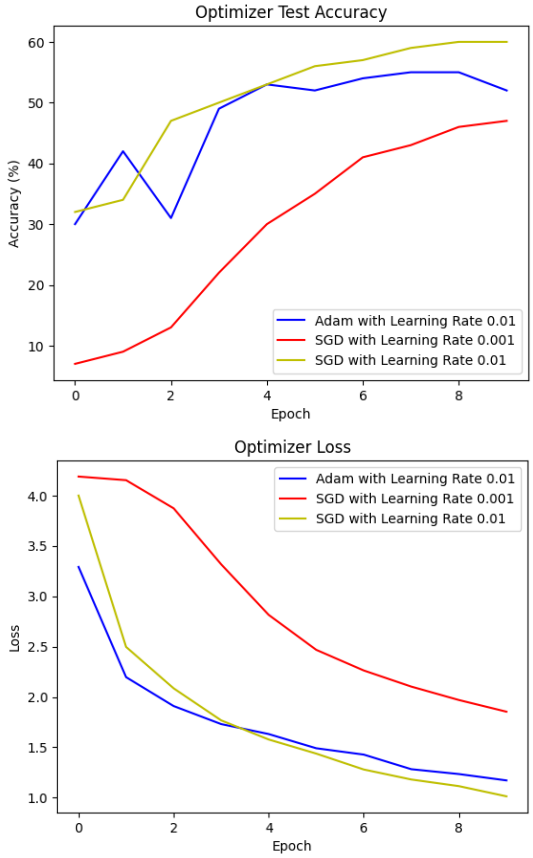

Figure 6: Learning rate test accuracy and loss.

Table III: Test accuracies using Adam.

Optimizer	Learning Rate	Test Accuracy
Adam	0.01	55%
SGD	0.001	47%
SGD	0.01	60%

Figure 6 displays the test accuracy and loss when comparing Adam to SGD.

Based on our results in Figure 6, we note that Stochastic Gradient Descent exhibited the highest test accuracy of 60% when compared to Adam. However, the loss function of Adam was significantly higher than models run with SGD in both test cases. A model of Adam with a learning rate of 0.001 was run. However, the model was unable to converge and thus terminated and excluded from this work. Given the measured similarity in test accuracy between SGD and Adam, there does not appear to be any significant discrepancy between the optimization functions, with SGD only slightly outperforming Adam on this test.

D. Implementation of Leaky ReLU

In this set of experiments, we substitute the usage of ReLU in AlexNet with Leaky ReLU. Mathematically, Leaky ReLU can be represented using Equation 2 [12]. Here, α represents a small value added in instances where x is negative. In practice, it has been shown that LeakyReLU generates better performance than ReLU.

$$f(x) = max(\alpha x, x) \tag{2}$$

Figure 7 provides a graphical representation of ReLu, presented as a), versus leaky ReLU, presented as b) as an activation function.

Table IV displays the final test accuracies using Leaky ReLU.

Table IV: Test accuracies using Leaky ReLU

Leaky ReLU	Test Accuracy
Yes	61%
No	60%

Figure 8 displays the test accuracy and loss when comparing Leaky ReLU to ReLU with respect to epoch.

E. Implementation of Cross-validation

Here, we look to implement cross-validation into our CNN architecture. The main motivation behind this is to investigate our model's ability to classify new data. Additionally, cross-validation has been shown to limit cases of overfitting and selection bias [13].

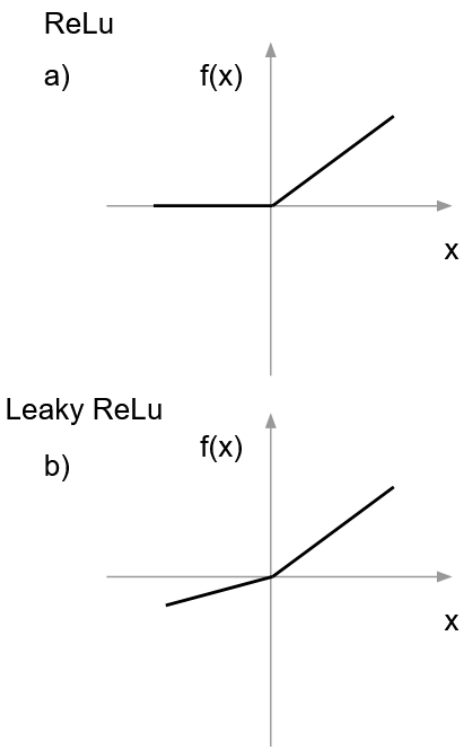

Figure 7: Graphical representation of the activation functions ReLU a) versus Leaky ReLU b).

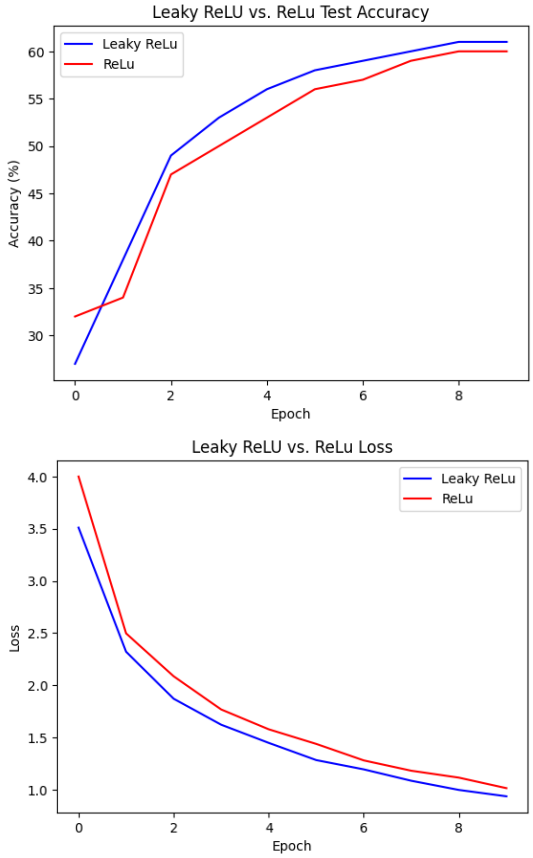

Figure 8: Learning rate test accuracy and loss with respect to using Leaky ReLU and ReLU.

Here, we perform 5-fold cross-validation on our dataset. We use the sklearn implementation of K-folds cross-validation and interfaced that with our CNN model. Table V displays the final test accuracies using 5-fold cross-validation.

Table V: Test accuracies using 5-fold cross-validation.

Cross-validation	Test Accuracy
Yes	65%
No	60%

Performing cross-fold validation marginally increased the accuracy of our model, only increasing it by 5% when compared to baseline. Given this increase, 5-fold validation would be useful when training a model for official deployment. However, the increased computational time can be fine-tuned and reduced through the use of GPUs to mitigate the costs of using this model.

F. Implementation of Data Augmentation

As discussed previously, the dataset used is an imbalanced classification problem. To attempt to solve this dilemma, we used data augmentation to artificially sample data. Specifically, we used three types of affine transformations on our data: rotation, translation, and scaling. Table VI displays the data augmentation parameters used.

Based off of the metrics reported in the VII, we note that applying data augmentation to the sample dataset results in a significant increase in test accuracy, increasing by 14%. This is likely due to the various configurations of plankton within the model, with the augmentation allowing the dataset to be more realistic through the application of random transformations. Particularly given the imbalanced dataset of this project, these transformations appear to make a significant difference in the performance of the model. The current parameter list shown in VI was based off of traditional values, but further testing would allow for these parameters to be fine-tuned.

Table VI: Description of affine transformations in data augmentation.

Affine Transformation	Parameter Range
Rotation	-20° to 20°
Translation	10% in the X and Y directions
Scaling	90% to 110%

Table VII: Test accuracies with and without data augmentation

Data Augmentation	Test Accuracy
Yes	74%
No	60%

IV. Conclusion

In this study, we have shown that CNN architectures are able to accurately classify a large sample of images. Using AlexNet, our group was able to achieve a maximal test accuracy of 74% using Adam as our optimization function and a learning rate of .01. Due to having extremely limited computational resources, more extensive models with larger layers were not able to be trained for this work, but future attempts at this project would like to leverage GPUs for more training. Additionally, we would like to be able to combine several of the different architectures that we note in this project, such as both Adam and LeakyReLU. As we noted in the comparison between SGD and Adam, Adam was outperformed in isolation by the SGD algorithm, but outperformed SGD in the final, best accuracy model. As such, we would like to conduct more extensive testing to optimize the model more consistently and determine which hyperparameters are more important to the predictive accuracy of the model. However from our results, data augmentation appears to make the most significant difference between models. Given the imbalanced dataset, resampling the data in order to balance our the number of images in each class could also allow for more accurate representation in the model space. Finally, we would like to perform visual inspection of those images that the algorithm confuses regularly, as shown in Figure 4. Inspection of these features may provide greater insight as to what preprocessing should be carried out on the data. Additionally, we would like to experiment with resizing and color correction, as the protocol implemented in this study was based on the original AlexNet architecture. However, we do believe that this architecture is able to accurately classify, and with increased testing and computation should be able to achieve higher levels of accuracy and achieve practical implementation in oceanography stations globally.

Acknowledgment

The authors of this paper would like to thank Professor ZhuoWen Tu, as well as the COGS 181 TAs, for their constant guidance and support throughout the quarter.

References

[1] Kefeng Li, Jane C. Naviaux, Sai Sachin Lingampelly, Lin Wang, Jonathan M. Monk, Claire M. Taylor, Clare Ostle, Sonia Batten, Robert K. Naviaux. Historical biomonitoring of pollution trends in the North Pacific using archived samples from the continuous plankton recorder survey. Science of The Total Environment, 2022; 161222.

[2] W. Rawat and Z. Wang, "Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review," Neural Computation, vol. 29, no. 9, pp. 2352–2449, Sep. 2017.

[3] N. Sharma, V. Jain, and A. Mishra, "An Analysis Of Convolutional Neural Networks For Image Classification," Procedia Computer Science, vol. 132, pp. 377–384, 2018.

[4] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA).

[5] L. Alzubaidi et al., "Review of deep learning: concepts, CNN architectures, challenges, applications, future directions," Journal of Big Data, vol. 8, no. 1, Mar. 2021.

[6] A. Dhillon and G. K. Verma, "Convolutional neural network: a review of models, methodologies and applications to object detection," Progress in Artificial Intelligence, vol. 9, no. 2, pp. 85–112, Dec. 2019.

[7] M. Z. Alom et al., "The History Began from AlexNet: A Comprehensive Survey on Deep Learning Approaches," arXiv:1803.01164 [cs], Sep. 2018.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, vol. 60, no. 6, pp. 84–90, May 2012.

[9] Agarap, Abien Fred, "Deep Learning using Rectified Linear Units (ReLU)," arXiv.org, 2018. https://arxiv.org/abs/1803.08375.

[10] "National Data Science Bowl," kaggle.com. https://www.kaggle.com/c/datasciencebowl

[11] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv.org, Dec. 22, 2014. https://arxiv.org/abs/1412.6980.

[12] A. Maas, A. Hannun, and A. Ng, "Rectifier Nonlinearities Improve Neural Network Acoustic Models."

[13] M. Stone, "Cross-validation: a review," Series Statistics, vol. 9, no. 1, pp. 127–139, Jan. 1978.