Fine-tuning a pre-trained Convolutional Neural Network Model to translate American Sign Language in Real-time

Manuel Eugenio Morocho Cayamcela
Department of Electronic Engineering
Kumoh National Institute of Technology
Gumi, South Korea
eugeniomorocho@kumoh.ac.kr

Wansu Lim
Department of IT Convergence
Kumoh National Institute of Technology
Gumi, South Korea
wansu.lim@kumoh.ac.kr

***Abstract*—In this paper, we present a real-time American Sign Language (ASL) hand gesture recognizer based on an artificial intelligence execution, instead of the classical and outdated image processing modalities. Our approach uses a Convolutional Neural Network (CNN) to train a dataset of hundreds of instances from the ASL alphabet, extracting the features from each and every pixel and constructing an accurate translator based on predictions. This approach employs an atypical trade-off for a translator, where a superior precision and speed at the inference phase compensates for the computational expense at the early training. Furthermore, and to the best of our knowledge, the accuracy obtained by using the proposed deep learning technique, surpass the accuracy obtained using non-machine learning practices. The performance obtained by the proposed algorithm has also been compared with existing literature, showing that the suggested methodology outperformed the accuracy of its analogous counterparts.**

***Index Terms*—Artificial Intelligence, Transfer Learning, Convolutional Neural Network, Image Classification, Sign Language, Real-Time.**

I. Introduction

The *National Institute of Deafness and other Communication Disorders* defines the *American Sign Language (ASL)* as a complete, complex language that employs hand signs to express ideas and communicate messages among people who are deaf or hard-of-hearing. [1].

From recent sign language recognizers proposed, some base their functionality in hard-code algorithms like edge detection [2], for instance. Moreover, these algorithms need pre-processing like image segmentation to remove unwanted data, or morphological filters to remove imperfections (dilation, erosion).

Some authors investigated the use of artificial intelligence-based algorithms to recognize sign language image examples. Islam et al. [3], used artificial neural networks (ANN) with feed forward, and back propagation for training with 30 feature vectors. They combined *K-curvature* and *convex hull* algorithms for feature extraction to detect the fingertip as a pre-processing stage, reaching an accuracy of 85.9% on the testing samples.

Pansare et al. [4], and Konwar et al. [5], proposed hand gesture recognizers of ASL based on edge orientation histograms (EOH), using a pre-processing stage that includes region and feature extraction, feature matching, and pattern recognition, accomplishing an accuracy of 88.26% and 65%, respectively.

Recent Artificial Intelligence (AI) developments like deep learning and transfer learning can assist in bridging this communication gap, while improving the classification accuracy and computational power needed at the inference phase, adding the value of automation, pattern recognition, feature extraction, and neural learning, reducing costs of software development and maintenance [6].

Other researchers have used datasets with an insufficient number of training samples for each alphabet class (letter), ranging from 10 instances in [2], 50 in [3], and 100 in [4], [5]. Using a small dataset leads to inaccurate accuracy measurements and models that do not generalize well on external data [7]. Our approach includes the use of a custom dataset with 3,000 training samples for each alphabet letter to make it more robust.

We proposed a new technique to enhance the accuracy from previous approaches by using a pre-trained CNN architecture, re-using the layers with the trained weights on feature extraction. This approach intends to improve the accuracy of actual alphabet sign language recognizers, rather than simply applying feature extraction and image pre-processing. The notion is to modify an existing state-of-the-art CNN by replacing the last set of fully-connected layers with our new set of fully-connected layers with randomly initialized weights. Our AI-based image classifier is built upon a custom dataset that takes into account various types of background, skin color, lightning conditions, the position of the hand within the camera box, and diverse types of hand sizes from different volunteers that collaborated with the research.

II. Classification Task using CNN

A. Dataset

The dataset is considered as the input component, from where the network will extract the features and learn to

978-1-5386-9223-3/19/$31.00 ©2019 IEEE

Fig. 1. Image instances of our generated dataset to feed the network, showing examples for the letter *"K"*. The dataset accounts for different backgrounds, lightning conditions, angles, positioning, skin colors, and hand sizes

discriminate the different classes, this data arrangement is usually denoted as *multi-dimensional design matrix* [8], that includes the raw pixel intensities from the images, and their associated class labels.

Considering our ASL dataset with 78,000 images in the RGB color space, each one has to be resized from $647x511x3$ to 227x227x3 pixels, the dimensions used to train AlexNet [9], and 224x224x3 pixels for GoogLeNet [10]. The design matrix from our dataset is represented as

$$\boldsymbol{X} \subseteq R^{Nx(w_i \text{ x } h_i \text{ x } c_i)} \tag{1}$$

with N as the total number of images in the dataset, and the dimension values of w_i, h_i, and c_i representing the width, height, and number of channels of the i-th image, respectively.

Assuming $i = 1, ..., N$ and $\boldsymbol{y_i} = 1, ..., K$, that means a dimensionality of N data points, corresponding to K different classes, where $\boldsymbol{X_i}$ represents the i-th image in R. The associated labels are contained in the vector $\boldsymbol{y}$, where each label is represented by $\boldsymbol{y_i}$. Fig 1. shows a subset of images from the class label *"K"*.

B. Scoring Function

The scoring function f maps the input data points to the class labels $f(\boldsymbol{X}) = \boldsymbol{y}$. A formal description of the scoring function f is described as a linear mapping as follows:

$$f(x_i, \boldsymbol{W}, b) = \boldsymbol{w}x_i + \boldsymbol{b} \tag{2}$$

This function will be used in *Section III* to train the weights of the network.

C. Loss Function

The loss function evaluates how well the predicted label of the class comply with the labels from the ground-truth data. If the agreement level is high, the loss on the training set will decrease. Our goal is to minimize the loss function (thereby increasing the accuracy) of the machine learning model.

D. Weights and Biases

Based on the output of the trained model, the parameters *weight matrix* $\boldsymbol{W}$ and *bias vector* $\boldsymbol{b}$ will be optimized in order to increase the classification accuracy. The bias vector can be used to translate or shift the scoring function influencing $\boldsymbol{W}$.

III. Classification Task and Algorithm

Each value of x_i is translated into a dimensionality vector with shape $[D, 1]$, where D is obtained by flattening the resized i-th image of (w_i x h_i x c_i) into a single column vector of value 154,587 for Alexnet, and 150,528 for GoogLeNet. The weight matrix $\boldsymbol{W}$ would then have a shape of $[K, D]$. Lastly, the bias vector $\boldsymbol{b}$ would be of size $[K, 1]$. Figure 2. shows an illustration of a linear classification process using the scoring function f.

The weight matrix $\boldsymbol{W}$ is composed of 26 rows representing the number of classes, and 154,587 columns, expressing the number of pixels in the image (including the 3 RGB channels). The arithmetic result of the dot product and the addition yields the scoring function, appending each value with a class label.

Therefore, parametrized learning has to be applied over our weight matrix $\boldsymbol{W}$, and our bias vector $\boldsymbol{b}$, by means of optimization methods such as gradient descent and its variants [11], [12].

A. Transfer Learning

Fine-tuning is applied to deep learning models that have been trained before on a different dataset. Our proposed approach is to remove the last set of fully-connected layers of one of these models, and replace them with our new set of fully-connected layers (with the same size as the number of classes in our new data) with random initializations along with our softmax classifier, freezing the rest of the network so their weight can not be updated when the errors propagate backwards [13]. Then, our network is training using a very small learning rate in order for the new fully-connected layer to start learning patterns from the previous convolutional layers in the first stage of the architecture. Applying this methodology, we fine-tune a pre-trained CNN to infer on new classes that they were not trained on, reaching a higher accuracy than the practices presented in *Section I*. The models used in this paper (Alexnet and GoogLeNet) are state-of-the-art architectures trained on the ImageNet dataset [14]. These networks contain rich, discriminative filters that will be reused. Figure 3. illustrates the changes made for Alexnet topology.

Transfer learning refers to the improvement of the generalization of our algorithm by exploiting what has been learned in one distribution setting P_1, into another distribution P_2. We assume that many factors that explain the variations in the

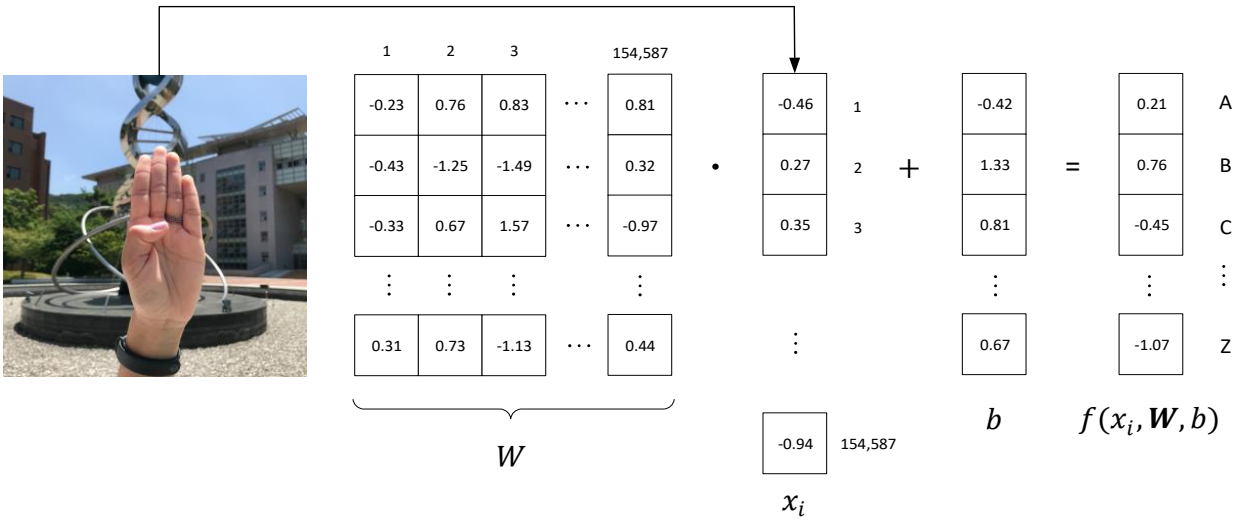


Fig. 2. Exemplification of the dot product between the weight matrix $\boldsymbol{W}$ and the feature vector $\boldsymbol{x}$, followed by the bias addition to flatten the example into a vector of 154,587 pixel intensities (in the Alexnet case) by taking the 3-dimensional array and reshaping it. The image shows the feature extraction $\boldsymbol{x_i}$ for letter "B" in the American Sign Language alphabet.

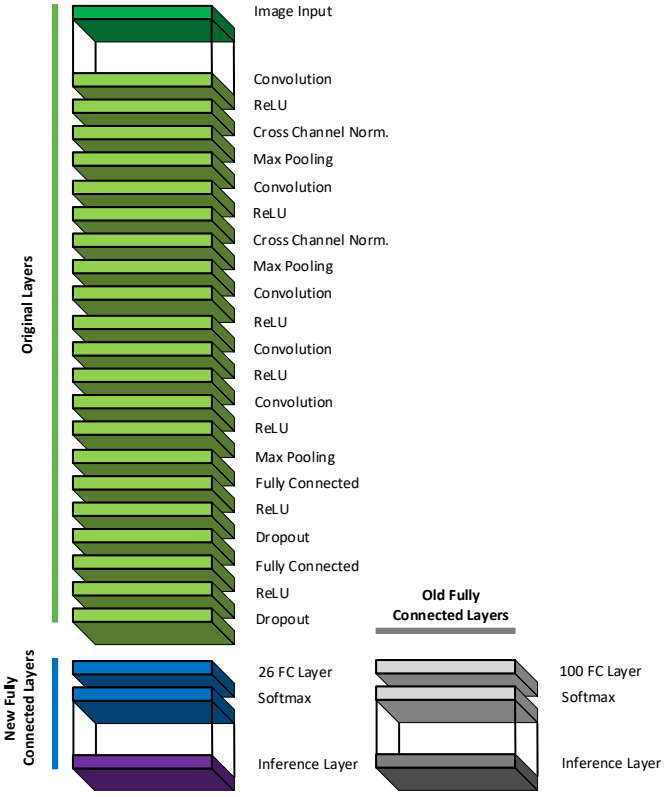


Fig. 3. Network architecture of Alexnet with its five convolutional layers and three fully connected layers. The bottom-left part shows the new layers added (26 fully connected layer, softmax activation function, and the new inference layer with our 26 letters of the alphabet) in order to reuse the architecture for our classification task. On the right, the old fully connected layer, softmax activation function, and classification layers are shown.

distribution P_1, are relevant to the changes that P_2 needs for learning. This technique will allow us to train a very robust algorithm with significantly less data than the first setting, and be able to generalize accurately [8].

Applying fine-tuning shows that our technique is an extremely powerful procedure, as an entire training of a network from scratch is not needed. This procedure has lead to higher accuracy, and using fewer examples than the required for an entire network training. Algorithm 1, exemplifies the procedure to build the sign language classifier using transfer learning.

Algorithm 1 Algorithm for Training the Neural Network

Input: Number of classes K
$[D, 1]$ vector of $\boldsymbol{x_i}$
$[K, D]$ weight matrix $\boldsymbol{W}$
$[K, 1]$ bias vector $\boldsymbol{b}$
Pre-trained neural network, number of classes
Output: retrained neural network
Initialisation :
1: compute $\boldsymbol{W x_i} + \boldsymbol{b}$
LOOP Process
2: **for** $i = 1$ to K **do**
3: Train network
4: Update weights from $\boldsymbol{W}$ and $\boldsymbol{b}$
5: **if** $(batch = max\_batch)$ **then**
6: finish training
7: **end if**
8: **end for**
9: **return** $f(input\_image)$

TABLE I
ACCURACY COMPARISON OF THE PROPOSED WORK WITH EXISTING LITERATURES

Method	Accuracy
Hue, Saturation, and Value model with edge detection [5]	65.00%
ANN with K curvature and Convex hull for feature extraction [3]	85.90%
Vision-based using edge-orientation histogram [4]	88.26%
Edge detection, morphological filtering, and cross correlation [2]	94.23%
Transfer Learning with GoogLeNet CNN	**95.52%**
Transfer Learning with Alexnet CNN	**99.39%**

IV. RESULTS AND PERFORMANCE COMPARISON

Fig 4. shows the results of the classification using the retrained Convolutional Neural Network on our dataset. As it is described in the previous sections, the key difference from our proposed method compared with traditional hard-code software is the flexibility of the classes to be recognized. If classes representing entire words needs to be classified, the algorithm will remain the same, and a simple update of the database with the new images to be recognized is required. The artificial intelligence methodology proposed will use the *learning* capability from the CNN and automatically extract the features from the new instances and update the weights and biases needed for the prediction.

Figure 5. shows the training process for our dataset on the transferred networks from Alexnet, and GoogLeNet, respectively. The figure shows the results of four full training cycles on the entire dataset, as one of the advantages of using transfer learning is the reduced number of epochs required to converge.

A performance comparison of analogous ASL translator approaches is shown in Table I.

Data augmentation was used on the training dataset, computing operations as a random reflection on the horizontal axis, translation on a 30-pixel range over the x and y-axis to help to prevent the network from overfitting and memorizing the exact details of the training images [15], [16]. A mini-batch size of 128 was established, as well as an initial rate of $1e^{-4}$, and a validation frequency of 50 learning iterations. The learning stage was executed in an environment of 4 CUDA® enabled graphical processing units (GPUs) *NVIDIA GeForce GTX 1080 Ti*.

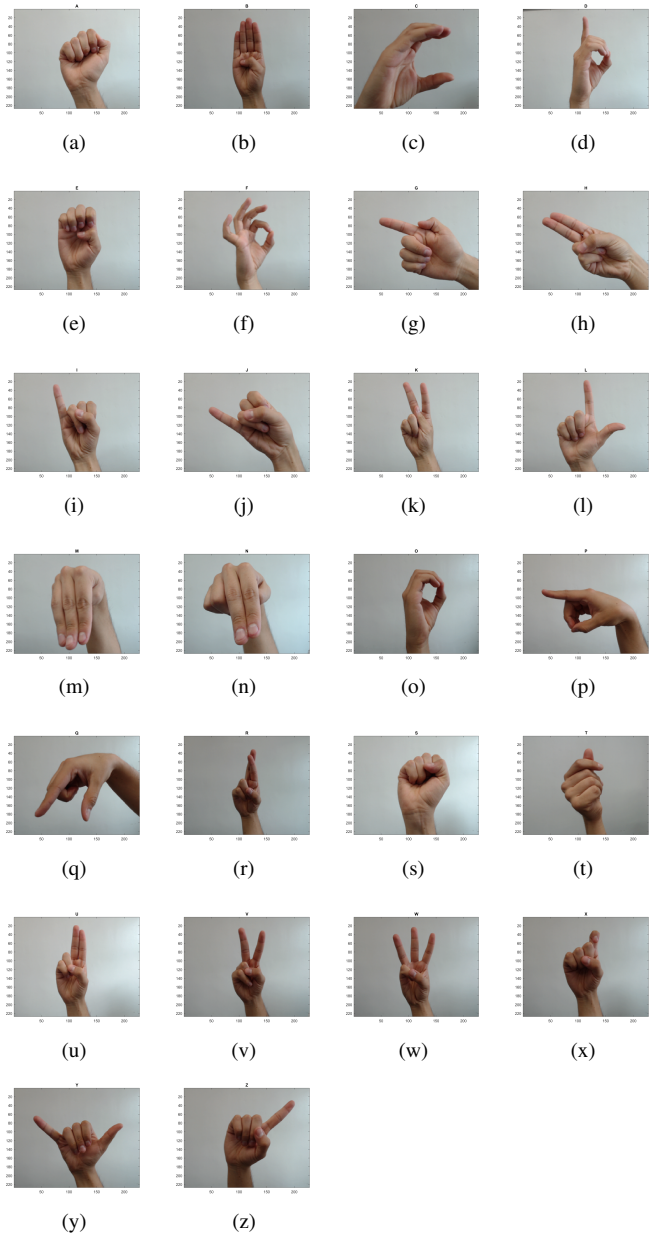
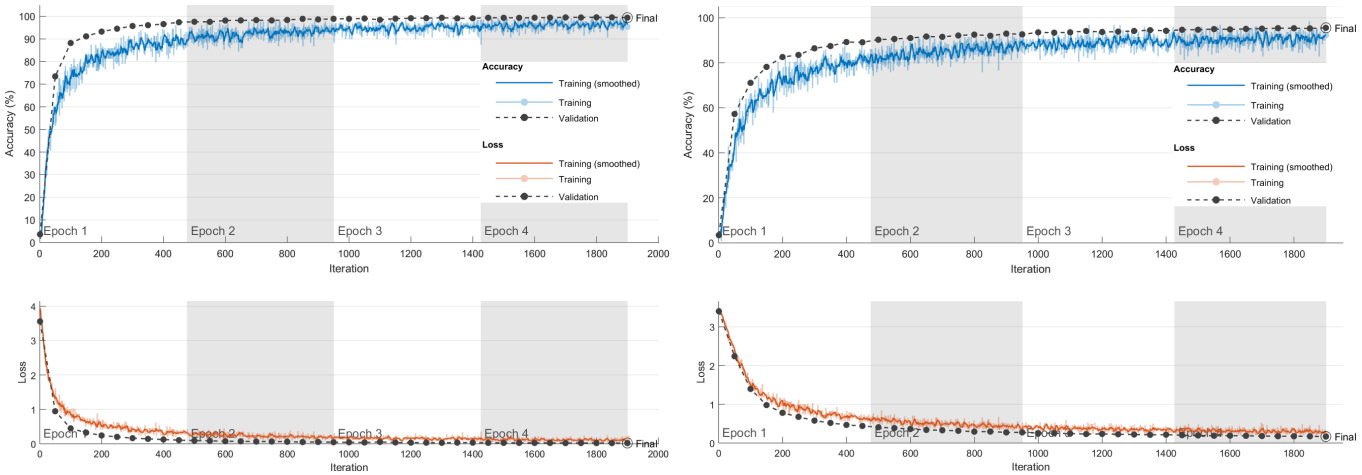


Fig. 4. The sub-captions (a)-(z), represents the results of the predictions according to the *American Sign Language* documentation after the retrained Convolutional Neural Network run the inference in new data instances.

V. CONCLUSION AND FUTURE WORKS

Transfer learning and fine-tuning methodologies have been tested on an enhanced American sign language alphabet dataset, proving that it can achieve higher accuracy than traditional image pre-processing techniques. Our rebuilt network reached an accuracy of 95.52% with GoogLeNet, and 99.39% with Alexnet. The downside part is that fine-tuning requires to analyze in advance the choices in fully-connected head parameters, as they play a crucial part in the overall network accuracy. Also, we can't rely strictly on regularization techniques as the initial CNN has already been pre-trained and deviation of the already performed regularization is restricted. However, while this fine-tuning procedure requires some effort while choosing the new layers to replace, if it is done correctly, a higher accuracy is nearly always guaranteed.

Future plans for this research includes testing a recursive neural network (RNN) in a sequence of images in order to recognize sign language as a progression, something closer to natural-language processing [17], and retraining CNN based on capsule networks [18].



(a) Case I Alexnet

(b) Case II GoogLeNet

Fig. 5. Training process for our dataset over the two different convolutional neural networks with the same initialization parameters. (a) Alexnet. (b) GoogLeNet. The graph shows a faster convergence using Alexnet than using GoogLeNet as a base of our new model.

ACKNOWLEDGEMENT

This work was supported by the Global Excellent Technology Innovation Program (10063078) funded by the Ministry of Trade, Industry and Energy (MOTIE) of Korea; the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP; Ministry of Science, ICT & Future Planning) (No. 2017R1C1B5016837); the "Leaders in INdustry-university Cooperation +" Project, supported by the Ministry of Education and National Research Foundation of Korea; and by the Research Fund, Kumoh National Institute of Technology.

REFERENCES

[1] National Institute on Deafness and Other Communication Disorders, "American Sign Language," pp. 2–5, 2015. [Online]. Available: https://www.nidcd.nih.gov/health/american-sign-language

[2] A. Joshi, H. Sierra, and E. Arzuaga, "American sign language translation using edge detection and cross correlation," in *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*. IEEE, 8 2017, pp. 1–6. [Online]. Available: http://ieeexplore.ieee.org/document/8088212/

[3] M. Islam, S. Siddiqua, and J. Afnan, "Real Time Hand Gesture Recognition Using Different Algorithms Based on American Sign Language," in *2017 IEEE International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*, 2017.

[4] J. Pansare and M. Ingle, "Vision-Based Approach for American Sign Language Recognition Using Edge Orientation Histogram," in *2016 International Conference on Image, Vision and Computing*, 2016, pp. 86–90.

[5] A. S. Konwar, B. S. Borah, and C. T. Tuithung, "An American Sign Language detection system using HSV color model and edge detection," *2014 International Conference on Communication and Signal Processing*, pp. 743–747, 2014. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6949942

[6] Y. A. LeCun, Y. Bengio, and G. E. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. [Online]. Available: https://www.nature.com/articles/nature14539.pdf

[7] A. Geron, "Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems," p. 543, 2017. [Online]. Available: http://shop.oreilly.com/product/0636920052289.do

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, 1st ed., T. Dietterich, Ed. London, England: The MIT Press, 2016. [Online]. Available: www.deeplearningbook.org

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks." [Online]. Available: https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, 2015, pp. 1–9. [Online]. Available: https://arxiv.org/pdf/1409.4842.pdf

[11] A. Rosebrock, *Deep Learning for Computer Vision with Python (Starter Bundle)*, 1st ed. Baltimore, Maryland: PyImageSearch, 2017. [Online]. Available: https://www.pyimagesearch.com/deep-learning-computer-vision-python-book/

[12] G. Villarrubia, J. F. De Paz, P. Chamoso, and F. D. la Prieta, "Artificial neural networks used in optimization problems," *Neurocomputing*, vol. 272, pp. 10–16, 2018.

[13] D. Rumelhart, G. Hinton, and R. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986. [Online]. Available: https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop_old.pdf

[14] P. U. Stanford Vision Lab, Stanford University, "ImageNet." [Online]. Available: http://image-net.org/about-overview

[15] MathWorks, "Transfer Learning Using AlexNet - MATLAB Simulink - MathWorks United Kingdom," 2018. [Online]. Available: https://uk.mathworks.com/help/nnet/examples/transfer-learning-using-alexnet.html

[16] Mathworks, "Transfer Learning Using GoogLeNet - MATLAB Simulink - MathWorks United Kingdom," 2018. [Online]. Available: https://uk.mathworks.com/help/nnet/examples/transfer-learning-using-googlenet.html

[17] X. R. S. S. J. He, Kaiming; Zhang, "Deep Residual Learning for Image Steganalysis," *Multimedia Tools and Applications*, pp. 1–17, 2017.

[18] S. Sabour, N. Frosst, G. E. Hinton, and G. B. Toronto, "Dynamic Routing Between Capsules." [Online]. Available: https://arxiv.org/pdf/1710.09829v1.pdf