Ghanaian Sign Language Recognition Using Deep Learning

Lamptey K. Odartey
Donghua University
Shanghai, China
+8618621349556
lampteyk@mail.dhu.edu.cn

Yonfeng Huang
Donghua University
Shanghai, China
+862167792165
yfhuang@dhu.edu.cn

Effah E. Asantewaa
Donghua University
Shanghai, China
+8618601675131
esthereffa@gmail.com

Promise R. Agbedanu
Donghua University
Shanghai, China
+8618616070405
ricardopromise@gmail.com

ABSTRACT

Sign Languages, unlike natural languages, involve the use of continuous gestures, body languages, facial expressions and hand movements to convey meaning and most importantly express a signer's thoughts more effectively. Ghanaian Sign Language is the standard sign language used by the deaf in Ghana with a substantial difference to other sign languages as well as cultural conditions that led to its emergence. In this paper, we proposed and implemented a novel yet deep convolutional neural network to classify and recognize Ghanaian Sign Language and attained an accuracy of 96.0%. Further, we leveraged transfer learning techniques by fine-tuning state-of-the-art network architectures pre-trained on the ImageNet database and improved the accuracy with a reported increase of 3.1%. There was no large publicly Ghanaian Sign Language dataset available, so we created our own dataset for evaluation of the proposed convolutional neural network architecture. Conclusively, we plan of extending the dataset with a view of releasing it in the future, subsequently, allowing researches to apply changes to the dataset using image processing and computer vision tools and techniques they consider can be applicable for their task at hand.

CCS Concepts

• Computing Methodologies → Machine learning → Learning paradigms → Supervised learning → Supervised learning by classification • Artificial intelligence → Computer Vision → Computer Vision problems → Object recognition • Computing methodologies → Machine learning → Machine learning approaches → Neural networks • Machine learning → Learning paradigms → Multi-task learning → Transfer learning

Keywords

Ghanaian Sign Language, Computer Vision, Convolutional Neural Network, Deep Learning, Transfer Learning

1. INTRODUCTION

Sign Languages, unlike natural languages, involve the use of continuous gestures, body languages, facial expressions and hand movements to convey meaning and most importantly express a signer's thoughts more effectively. Sign languages are a form of communication for the deaf. Also, it provides a way to communicate without the use of voice.

This is very important for individuals who have impaired hearing and speech disability as a result of the medium of communication it provides between the speech-impaired and the hearing community. Ghanaian Sign Language (GSL) is the standard sign language used by the deaf in Ghana with a substantial difference to other sign languages as well as cultural conditions that led to its emergence. Ghanaian Sign Language is a full-fledged human language and has phonology and morphology which is descriptive and analyzable like any other documented language (spoken and signed) [1].

Research in sign language recognition has upsurged due to the viability of deep learning in the computer vision field. There has been a lot of research on sign languages such as Indian Sign Language, American Sign Language, Italian Sign Language, Bangla Sign Language and many other sign languages, which has been very instrumental in the development of Sign Language Recognition (SLR) algorithms. Sign language differs for each country and research development in sign language has shown a significant increase in the past decade. However, much research has not been done with Ghanaian Sign Language (GSL) when it comes to leveraging computer vision and deep learning techniques to perform such tasks. In addition, the pedagogical approach of teaching sign language is daunting and discouraging to learners. Subsequently, making it difficult for the deaf community to communicate with members of the hearing community. To address this, we proposed and implemented a novel yet deep convolutional neural network to classify and recognize Ghanaian Sign Language. Further, we leveraged transfer learning techniques by fine-tuning state-of-the-art network architectures pre-trained on the ImageNet database [2].

The organization of this paper is as follows: Section 2 discusses related works with other sign languages. Section 3 presents the methodology. Section 4 explains the experimental set up followed by the results obtained from the experiment in section 5. Finally, we conclude the paper in section 6.

2. RELATED WORK

Prior to this paper writing, there is no related work regarding recognition of Ghanaian Sign Language using deep learning and computer vision. Without regard to, different approaches have been applied to the problem of recognition of other sign languages.

Shirin et al. [3] published a paper on using Convolutional Neural Network (CNN) and Scale-Invariant Feature Transform (SIFT) to classify Bangla sign language. The authors captured images manually and applied preprocessing techniques SIFT, K-means clustering and Bag of Features (BoF) to extract features for each

PRAI '19, August 26–28, 2019, Wenzhou, China

ACM ISBN 978-1-4503-7231-2/19/08…$15.00
https://doi.org/10.1145/3357777.3357784

class. The BoF represent each image as a histogram of the feature before used as input for the CNN model. They described that using SIFT, K-means clustering and BoF as pre-processing techniques before feeding the data into the CNN model can improve the performance of their model. Based on their findings, combing SIFT with CNN further improved the accuracy for each class of the sign language dataset. However, the drawback of this approach was that the images were converted to grayscale before fed into the CNN model. Subsequently, this will make the model fail to generalize well when used in real-time.

Rangel et al. [4] presented an approach to real-time sign language alphabet recognition using deep learning by fine-tuning the Dense Convolutional Network (DenseNet) [5] architecture. Rangel et al. approached the problem of classifying the sign languages by using four dense blocks of the DenseNet architecture together with three transition layers and using softmax activation as output to predict the letter of each class and achieved an accuracy of 90.3% on the test dataset. They could have further improved their model accuracy had they segmented the Region of Interest (ROI) from each class of image despite the possibility that they included faces of the signers to improve the robustness of the network by introducing some noise to the data.

Manuel et al. [6] proposed a new technique to enhance the accuracy of previous approaches to recognizing American Sign Language by using state-of-the-art CNN architectures GoogleNet [7] and AlexNet [8] pre-trained on the ImageNet dataset. Their approach of fine-tuning AlexNet and GoogleNet showed good preliminary results while using fewer training data for the network training. They obtained an accuracy of 95.52% with GoogleNet and 99.39% with AlexNet. Subsequently, proving that transfer learning and fine-tuning can achieve higher accuracy than traditional image pre-processing techniques, let alone training a CNN architecture from scratch.

Sajanraj et al. [9] used a Region of interest convolutional neural network for real-time recognition of Indian Sign Language. Sajanraj et al. approach to preprocessing involve using Contrast Limited Adaptive Histogram Equalization (CLACHE) on the image with the intention of equalizing the lightness in the image using the LAB colour system. Thresholding operation is then applied using the HSV colour space to obtain the skin after blurring the original image. They proposed a region of interest predictor using the skin segmentation. The image is then cropped from the segmented bounded region and then used as input before fed into their proposed region of interest convolutional neural network. The network was trained on only numerals of the Indian Sign Language captured using RGB camera and they obtained an accuracy of 99.56% during testing. However, the accuracy of the model dropped when tested in low light conditions.

3. METHODOLOGY

3.1 Dataset

To the best of our knowledge, there are no publicly available datasets for Ghanaian Sign Language. Due to the non-availability of a GSL dataset, we created a dataset based on the Ghanaian Sign Language second edition dictionary [10] to train our convolutional neural network and evaluate its performance. The Ghanaian Sign Language dataset consists of 66000 images in the RGB colour space, with 33 classes of static gestures consisting of 24 alphabets and 9 digits (1-9). 48000 samples out of the total sample of the static gestures are alphabets with the remaining 18000 samples as digits. Each class instance contains 2000 Ghanaian Sign Language static gestures. The dataset was further split into training, validation and test set of 50%, 25% and 25% respectively. This makes our dataset more robust as the training will be done on the split ratio of the training and validation data. Two letters (J, Z) and the other digits of GSL were excluded from the dataset due to the motion of gestures required to represent them.

We captured static gestures of images of alphabets and digits from a web camera under different illuminations and controlled background using the OpenCV image processing module [11]. We used more variations when performing the gesture for each sign, presenting a lot of challenges to any convolutional neural network model for classification. The variation for each image was based on setting the colour space and threshold values of Hue, Saturation, Value (HSV) track bars. The HSV track bars calibrator was implemented using the OpenCV module. This resulted in changing the background of the image to a colour space based on adjusting the values of HSV colour space calibrator. With our dataset, we randomly changed some of the images to black background by setting the lower saturation value to any value from 40 - 65 as well as the upper value of the HSV calibrator to any value from 212 – 230. Randomly setting the background of some of the images to black helped us to isolate more colours in the image. In addition, for every channel of the input array image, we checked if the array element lies in the region of interest values of the lower and upper HSV calibrator. Further, we computed the per-element bit-wise conjunction by processing each channel independently of the lower and upper HSV value. The image was then resized and saved to a uniform dimension of 512 * 512 pixels with varying intensities after cropping the region of interest. This reduces the overall file size, that is, the segmented hand gesture from the full web camera image. The region of interest was only captured from the full web camera image with the intention of preventing our model from learning irrelevant features. Procedure for the image capturing is illustrated in Algorithm 1.

$I_{w \times h \times c}$ represents a frame captured from the web camera. The subscript w represents the width, h as the height and c represent the channel.

Algorithm 1: Image Capturing Procedure

Input: Captured Frame $I_{w \times h \times c}$
Begin
1: Read $I_{w \times h \times c}$
2: Convert $I_{w \times h \times c}$ to HSV
3: Set upper and lower HSV values
4: Compute per-element bit-wise conjunction
5: Process each channel independently
6: Resize image to the target dimension
7: Save image
End

3.2 Convolutional Neural Network

A convolutional neural network is a deep artificial neural network algorithm which takes an image as input, learns spatial and temporal features anywhere in the input image and classifies an image based on the features learnt. We preferred using convolutional neural network for the proposed model in light of the fact that it requires much lower pre-processing on the dataset as compared to other shallow networks and classification algorithms. The temporal and spatial feature representations learnt by the convolutional neural network reflects the similarity among each class of the dataset. The network was trained to learn feature representations on all samples of the dataset. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a

good prediction [12]. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights [12]. The output of a convolutional layer is shown in the equation below:

$$y_j^n = f\left(\sum_{i \in c_j} y_i^{n-1} * k_{ij}^n + b_j^n\right) \quad (1)$$

Where y_j^n represents the output of the nth layer, n is the nth layer of the convolutional neural network, k_{ij} is the convolutional kernel, c_j is the input maps and b_j represents the bias.

3.3 Network Architecture

Our proposed CNN architecture consists of an input layer, four convolutional layers, four rectified linear unit (ReLu) and max-pooling for each layer, a dense layer and a SoftMax output layer. The four convolutional layers use a kernel size of $3\,X\,3$ while doubling the filter size for each layer based on the filter size used for the first convolutional layer. For instance, with a filter size of 32 for the first layer, subsequent layers get a filter size of $N \times 2^i$ where N represents the filter size used for the first convolutional layer and i is the number associated with the next convolutional layer. The classification stage is implemented using the fully connected layers followed by the softmax regression for classification. Each image is resized to a target size of 150×150 before fed into the model. Performing extensive experiments on our dataset using our novel deep convolutional neural network achieves accuracy and performance almost as good as the state-of-the-art fine-tuned architectures. Architecture of the proposed model is shown in the figure below.

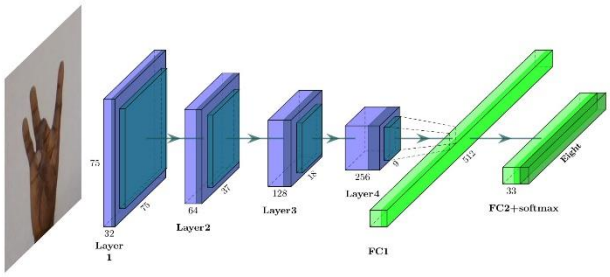
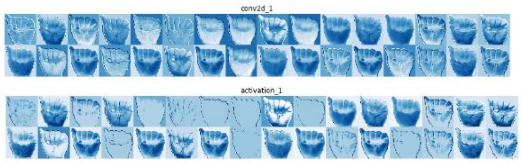


Figure 1. Proposed CNN architecture for Ghanaian Sign Language recognition

Figure 1. depicts how an image with a true label of ***Eight*** is fed into the proposed model. The convolutional neural network architecture uses the knowledge obtained from the features of each class of image during training to predict the class of the input image. Visualization of the activation of some layers of the network is shown in **Figure 2**.



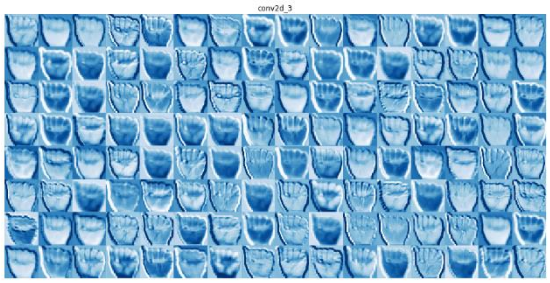


Figure 2. Visualization of the conv2D and ReLu activation function of the network on class A.

The figure above shows the activations of the first convolutional layer, third convolutional layer and the ReLu activation function of the network on class A. The first convolutional layer uses filter size of 32 whereas the third convolutional layer uses a filter size of 128. Both layers use a kernel size of 3×3. The ReLu activation function detects edges in the network whereas the first conv2D still maintain all the features in the initial input image. The model starts to encode higher level feature representations in the third convolutional layer.

4. EXPERIMENTAL SETUP

Details of our experimental set up implemented to attain the stated baseline results is presented in this section.

4.1 Data Augmentation

To improve the generalization of our model, we implemented an efficient data augmentation strategy by employing online data augmentation techniques with the intention of reducing overfitting, subsequently increasing the size and variety of the training set as well. The augmentation on both the training and the test set was done by applying rescaling, zooming, horizontal translation, and rotation randomly on the training set before being fed into the proposed network architecture.

4.2 Training

We represented the image as a three-dimensional array using last channel ordering as input to the proposed model after augmenting the images in the dataset. The training and testing were done in three-fold with both our proposed model and the fine-tuned state-of-the-art networks. The first group of the experiment was to classify all 33 classes of the dataset, namely, the alphabets and digits. Also, we trained both the proposed model and the fine-tuned networks to classify only the 24 classes of alphabets in the dataset. Finally, we trained each architecture to classify the remaining 9 classes of digits in the dataset.

Throughout each set of training and testing, we adopted the same batch size, channel order, loss function, optimization algorithm, metrics and augmentation techniques. A batch size of 32 was used during each set of training. For the proposed model, the number of epochs was fixed at 30 for each training and the best validation accuracy for an epoch was selected and used to evaluate the performance of the model. Similarly, the number of epochs for the fine-tuned models was fixed at 20, selecting the best validation accuracy during training for evaluating the performance of each fine-tuned architecture.

4.3 Loss Function

Categorical cross-entropy, also called softmax loss, was used as the loss function for our single label categorization. Using this loss function trains a convolutional neural network to output a probability over the number of classes for each image. This loss function compares the distribution of each of the activations in the output layer with the true distributions. The probability of the true class predicted is then set to 1 with the remaining classes set to 0.

4.4 Loss Function Optimization

Adam optimization algorithm was used to optimize the loss function during training with a learning rate of 0.001 and 0.000001 for our proposed model and the fine-tuned architectures respectively, at the same time, using the default β_1=0.9 and β_2=0.999 parameter values of the keras [13] module. Hyperparameter values used during training is shown in **Table 1**. Adam optimization algorithm is an extension to the stochastic gradient descent as it requires less memory, remarkably efficient with computations and reaches a global minimum when minimizing the cost function during the training of a neural network.

Table 1. Hyperparameter values used during training

Architecture	Epoch	Learning rate
Fine-tuned VGG-16	20	0.000001
Fine-tuned VGG-19	20	0.000001
Ours	30	0.001

4.5 Transfer Learning

Transfer learning is one of the techniques used in deep learning by transferring the knowledge learnt from a model on a dataset to a different dataset. In transfer learning, we first train a base network on a base dataset and task, and then we repurpose the learned features, or transfer them, to a second target network to be trained on a target dataset and task [14]. This process will tend to work if the features are general, meaning suitable to both base and target tasks, instead of specific to the base task [14].

We fine-tuned the VGG-16 and VGG-19 [15] networks and retrained both networks on our dataset by freezing the weights of the first few layers coupled with replacing the top layers and softmax layers of each network with fully connected layers and a new softmax layer that is relevant to the number of classes of our dataset. We used a smaller initial learning rate as it is fitting to use a very small learning rate with the view of allowing the network to learn the optimal set of weights during training. After each set of training of both networks, we attained a high-end accuracy as shown in **Table 3**, consequently improving the accuracy of our dataset.

4.6 Hardware and Software

Images were captured using an HP Wide Vision HD Camera with a resolution of 1280×720, MegaPixels of 0.92 MP, a frame rate of 30 FPS, 12.38 MB/s bit rate, and an aspect ratio of 1.78. The proposed model was developed in python using the keras open-source framework with tensorflow [16] as backend. We trained the proposed system on the Dell Precision Tower 3620 series running on Ubuntu operating system with a 64GB RAM, Intel(R) Core (TM) i7-7700 processor and a clock speed of @3.60GHz speed.

5. RESULTS

Our convolutional neural network coupled with the fine-tuned state-of-the-art architectures VGG-16 and VGG-19 were trained and evaluated independently on the test set for each set of class. The accuracy of the proposed model and the fine-tuned networks were tested on data it did not see during training with total images of 16500 for the 33 classes, 12000 for the 24 classes and 4500 for the remaining 9 classes of digits. The best-saved weight of the validation accuracy obtained during training was used in evaluating and measuring the performance of the network architectures on the test data using the classification accuracy and confusion matrix metrics. The confusion matrix is shown in **Figure 4**. The classification accuracy metric computes the ratio of the number of correct predictions to the total number of predictions made by the model. The accuracy attained by the proposed model and transfer learning with the state-of-the-art architectures is shown in the table below.

Table 2. Accuracy comparison with varying sample classes

Architecture	Accuracy		
	Total	24	9
Fine-tuned VGG-16	**99.1%**	**100%**	**98.0%**
Fine-tuned VGG-19	99.0%	**100%**	**98.0%**
Ours	96.0%	98.0%	92.0%

From the above table, the highest attained accuracy for the first experiment was 99.1%, 100% for the second experiment and 98.0% for the last set of experiment. Both fine-tuned networks obtained almost the same accuracy for each set of class and reported an increase of 3.1%, 2% and 6% for the 33 classes, 24 classes and 9 classes respectively. In addition, we obtained a high-end accuracy for the fine-tuned networks given that we freeze some of the layers pre-trained on the ImageNet dataset during training and retrain our classifier on top of it. Training process of the proposed architecture for the total classes of the dataset with less amount of overfitting is shown in the figure below.

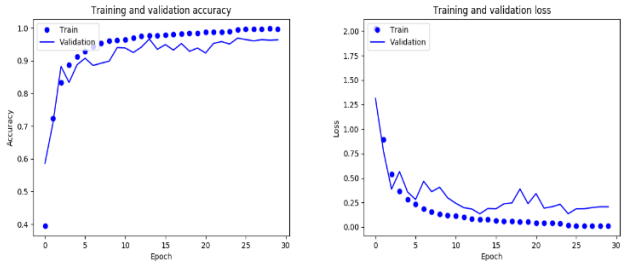


Figure 3. Training, validation accuracy and loss

In the figure above, the graph on the left displays the training and validation accuracies obtained at each epoch in the course of training the proposed model on the dataset. Also, the other graph displays the training and validation losses at each epoch of training the proposed model on the dataset. There is less amount of overfitting on both accuracies and losses even though there was no random dropout of nodes during training. This could be as a result of validating the model during training and setting the hyperparameter values of the network based on validation accuracy attained after each epoch. This helps in improving the

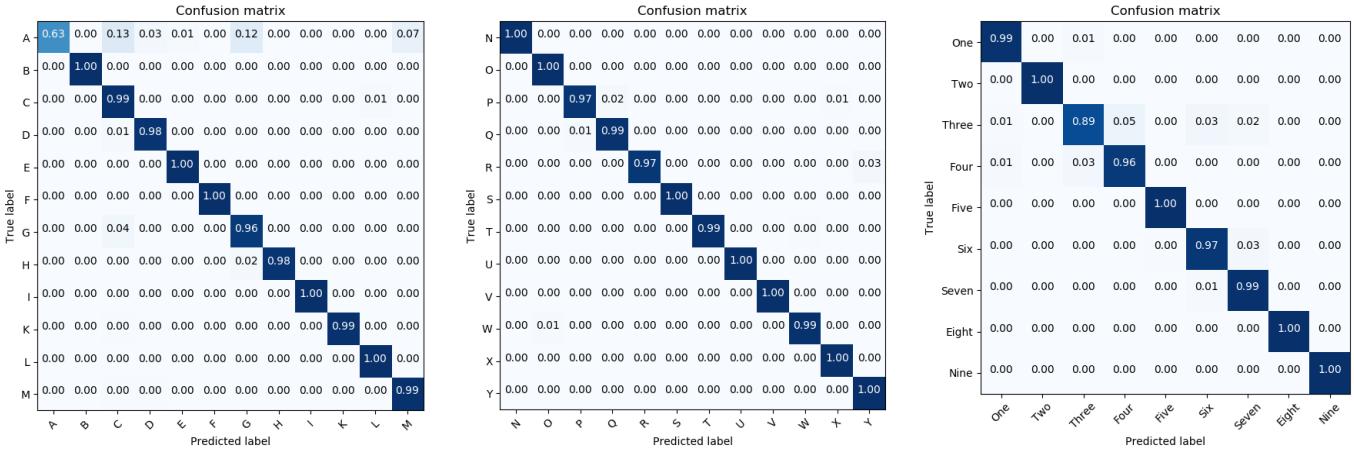


Figure 4. Confusion matrix for the total class

generalization of the model as well. A summary of the training results is displayed in **Table 3**.

Table 3. Summary of training results

Architecture	Epoch	Training		Validation	
		Acc.	Loss	Acc.	Loss
Fine-tuned VGG-16	20	99.3%	0.026	96.9%	0.210
Fine-tuned VGG-19	20	99.6%	0.017	**97.9%**	**0.052**
Ours	30	**99.7%**	**0.010**	96.4%	0.210

In **Table 3**, the figures present only the reported final training and validation accuracies attained during the training of the proposed model on the total classes. The best-saved weight of the validation accuracy obtained during training was used in measuring the performance of the network architectures on the test data. From the results shown in the table above, the proposed model attained the highest accuracy with the lowest training loss on the training data while transfer learning with VGG-19 network obtained the highest validation accuracy and lowest validation loss on the validation data. The validation accuracies and losses for the best-saved weights are shown in **Table 4**.

Table 4. Highest validation accuracy and loss results

Architecture	Epoch	Validation	
		Acc.	Loss
Fine-tuned VGG-16	**11**	98.5%	0.073
Fine-tuned VGG-19	14	**99.2%**	**0.027**
Ours	25	96.9%	0.206

From the table above, in the course of training the proposed model on the dataset, the best weight was obtained at epoch 25 with a validation accuracy and loss of 96.9% and 0.206 respectively. With the fine-tuned architectures, a validation accuracy and loss of 98.5% and 0.073 for VGG-16 was attained at epoch 11 whereas, at epoch 14, a validation accuracy of 99.2% and a loss of 0.027 was attained for VGG-19. Transfer learning with VGG-19 attained the highest validation accuracy rate. This exemplifies the reduced number of epochs required for training a classifier on top of a pre-trained network to converge in the course of training.

In **Figure 4**, it can be observed that the proposed model predicted a larger part of the classes accurately devoid of any confusion with other classes of the dataset. However, the model had difficulties predicting the letter A. Class A is confused with quite a few numbers of the classes C, G and M. This could be as a result of the online augmentation techniques implemented during training leading to the similarity in the abstract representations and features learnt by the network. Overall the confusion matrix yielded an average F1-score of 96%, precision of 96% and a recall of 96%.

6. CONCLUSION

The viability of deep learning in the computer vision field has upsurged research development of sign language recognition. However, prior to this work, no work had been done regarding leveraging deep learning and computer vision techniques for Ghanaian Sign Language recognition. In this paper, we proposed and implemented a novel yet deep convolutional neural network for Ghanaian Sign Language recognition and attained an incredibly high-end accuracy for our model. In addition, we improved the accuracy further by fine-tuning state-of-the-art networks VGG-16 and VGG-19 with a reported increase of 3.1%. There was no large publicly Ghanaian Sign Language dataset available, so we created our own dataset for evaluation of the proposed convolutional neural network architecture. In addition, we plan of extending the dataset with a view of releasing it in the future. Subsequently, allowing researches to apply changes to the dataset using image processing and computer vision tools and techniques they consider can be applicable for their task at hand.

7. REFERENCES

[1] Edward, Mary. 2014. The Phonology and Morphology of Ghanaian Sign Language. *First International Workshop on African Sign Language.*

[2] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *Proceedings of IEEE Conference on Computer*

Vision and Pattern Recognition (CVPR '09). Miami, FL, 248-255. DOI: 10.1109/CVPR.2009.5206848.

[3] S. S. Shanta, S. T. Anwar and M. R. Kabir. 2018. " Sign Language Detection Using SIFT and CNN. In *Proceedings of 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT '18)*. Bangalore, 1-6. DOI: 10.1109/ICCCNT.2018.8493915.

[4] R. Daroya, D. Peralta, P. N. Jr. 2018. Alphabet Sign Language Image Classification Using Deep Learning. In Proc. IEEE Region 10 Conference, TENCON 2018".

[5] G. Huang, Z. Liu, K. Q. Weinberger, and L. van der Maaten. 2017."Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '17)*. 3.

[6] M. E. M. Cayamcella, W. Lim. 2019. Fine-Tuning a pre-trained Convolutional Neural Network to translate American Sign Language in Real-Time. In *Proceedings of International Conference on Computing, Networking and Communications (ICNC '19)*. Honolulu, HI, 100-104. DOI: 10.1109/ICCNC.2019.8685536.

[7] Szegedy, C. Liu, W. Jia, Y. Sermanet, P. Reed, S. Anguelov, D. Erhan, D. Vanhoucke, V., and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of International Conference on Computer Vision and Pattern Recognition (CVPR '15)*. Boston, MA, 1-9. DOI: 10.1109/CVPR.2015.729859.

[8] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2017. ImageNet classification with deep convolutional neural networks. Commun. ACM 60, 6 (May 2017), 84-90. DOI: https://doi.org/10.1145/3065386.

[9] Sajanraj. T D, Beena M. 2018. Using Region of Interest Convolutional Neural Network. In Proceedings of 2nd International Conference on Inventive Communication and Computational Technologies (ICICCT '18). Coimbatore, 636-640. DOI: 10.1109/ICICCT.2018.8473141.

[10] Deustch C, McGuire C. 2018. *Ghanaian Sign Language, Second Edition Dictionary* (2nd. ed.).

[11] Bradski G. 2000. OpenCV. *Dr. Dobb's Journal of Software Tools*, 120, (Nov. 2000), 122-125.

[12] Sumit Saha. 2018. A Comprehensive Guide to Convolutional Neural Networks—the ELI5 way. (Dec. 2018). Retrieved April 17, 2019 from https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[13] Francis Chollet et al. 2015. Keras: The Python Deep Learning library. (2015). Retrieved March 23, 2018, https://github.com/fchollet/keras.keras.io.

[14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (NIPS'14), Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.), Vol. 2. MIT Press, Cambridge, MA, USA, 3320-3328.

[15] K. Simonyan and A. Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556. Retrieved from https://arxiv.org/abs/1409.1556, 2014.

[16] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2016. TensorFlow: a system for large-scale machine learning. In *Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation* (OSDI'16). USENIX Association, Berkeley, CA, USA, 265-283.

[17] Jason Brownlee. 2019. Loss and Loss Functions for Training Deep Learning Neural Networks. (Jan. 2019). Retrieved March 10, 2019, from https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks/.

.

.