Implementation of a Filipino Sign Language Recognition Model on a Raspberry Pi 3 Using CNNs and TensorFlow Lite

Isaiah Jassen Tupal[1], Melvin Cabatuan[1] and Michael Manguerra[2]
[1] Electronics and Computer Engineering Department, De La Salle University
[2] Manufacturing Engineering and Management Department, De La Salle University
**Corresponding Author: isaiah_tupal@dlsu.edu.ph*

Abstract: Models that perform sign language recognition (SLR) have the potential to bridge the gap between the deafmute, and those who communicate verbally. Most of the research in this field focuses on the language while leaving Filipino Sign Language understudied. Furthermore, the state-of-the-art models that perform SLR are spatially large and require powerful machines. This can limit other embedded devices, such as Raspberry Pi, from implementation. to run. With this in mind, this paper focuses on Filipino Sign language, and on making the models run on a resource-scarce device. The device is a Raspberry Pi 3, while the models are the three versions of MobileNet and EfficientNet. TensorFlow lite was used to convert these models to tinyML models called TFLite. The results of the training show that all models perform well in 29 categories (26 letters and 3 non-letters). After conversion, only EfficientNet and MobileNetV3 are usable. Nevertheless, both models have potential as EfficientNet is accurate with a score of 96.55%, while MobileNetV3 is fast clocking in at 182 seconds for categorizing 435 sign language images. With this in mind, It can be concluded that these models have the potential for real-life implementation on an embedded device. Further exploration of other models and devices is also encouraged for more efficient and accurate models.

Key Words: Filipino Sign Language, Deep Learning, TinyML, MobileNet, EfficientNet

1. INTRODUCTION

One of the most important aspects of human living is our ability to communicate with others. Such a capability lets humans form and develop relationships with each other. In addition to that, people rely on communication to spread valuable information to different parties. The most common method of communication with other people is verbal communication. This fact becomes a problem when other communities, such as the deafmute, cannot participate in this form of communication. Hence, they express their thoughts through other modes such as sign language

In sign language, a signer utilizes visual cues to convey information (Fischer, 2015). Sign language is not a homogeneous language as it can be varied depending on the region of the signer. Examples of these languages include American sign language (ASL), French sign language, Chinese sign language (CSL), and Filipino sign language (FSL). For the latter, this language has been understudied in all forms as only 65 publications are related to FSL as of 2021 (Notarte-Balanquit, 2021). This is even more concerning because the hearing loss incidence in the country is significant with 1 for every 6 Filipino suffering a form of hearing loss (Newall et al., 2020).

Hence, Sign language research must be conducted to bridge the gap between a significant deafmute population and those who communicate verbally. One notable research is Sign language research SLR. SLR is done via trained machine learning and deep learning models to predict the sign given by the signer in this field. Numerous SLR models have been developed by various researchers, many of which are accurate (Rastgoo et al., 2021). As for Filipino sign language, there have been several researchers that focus on this specific language when it comes to SLR. One method used Kinect to extract the data from the signer and then utilize that data for an SVM model with an accuracy of 95% (Oliva et al., 2018). Another utilized ResNet architecture which yielded an accuracy of 86.7% (Montefalcon et al., 2021). These models show the potential of FSL recognition for application, however, usual machine learning models run on modern machines. The Skeleton Aware Multi-modal sign language recognition model, which is the state-of-the-art time for sign language recognition, is more than a gigabyte and requires a powerful machine to train and run (Jiang et al., 2021).

Complex heavy architectures can bring raw accuracy, however, these kinds of models exclude devices where SLR can operate. Not all models have to run on powerful machines as the field of Tiny Machine learning has let compressed Architectures work on more resource-constrained machines such as mobile devices and edge devices such as the raspberry pi. Tiny Machine Learning or TinyML is the field of utilizing or optimizing machine learning for on-edge devices (Al-Qurishi et al., 2021). TinyML unlocks the potential of ML to be used for applications. One of these applications can be SLR. With this in mind, this study aims to implement a Filipino Sign Language Alphabet Recognition Model on a Raspberry Pi 3 device using Resource Efficient Models and TensorFlow-lite.

2. RELATED WORKS

2.1 Efficient CNN Models

Most Sign language models utilize convolutional neural networks (CNN) due to how they can derive features that provide descriptive information from raw data. CNN is good at image classification applications, however, its complexity when it comes to computation is higher than classic algorithms, making them not portable enough for some applications (Arredondo-Velázquez et al., 2020). Hence, some models aim for portability without sacrificing accuracy. One class of these models is the MobileNets. MobileNets are a class of models that are developed with efficiency in mind. It is a convolutional network that is made for computer vision applications on embedded devices (Howard et al., 2017). Another CNN that is made with constrained resources in mind for vision application is the EfficientNet. This model is scalable to the point that it can surpass some state-of-the-art models when it comes to accuracy (Tan & Le, 2020).

2.2 TinyML and TFLite

The idea behind TinyML is that it proposes ML features to be run on resource-scarce devices such as Microcontroller Units (MCUs). With this view, novel implementations and applications can be developed without the need for the cloud to shoulder the computational cost (Sanchez-Iborra & Skarmeta, 2020). One framework that can aid in developing TinyML models and applications is TensorFlow Lite (David et al., 2021). TensorFlow Lite or TFlite converts models into TFlite file format which can be run on some embedded devices. TensorFlow Lite uses quantization to make inference more efficient by making the calculation use integer-only arithmetic instead of floating-point one (Jacob et al., 2017).

3. METHODOLOGY

3.1 Training of Data

For the collection of data, the researchers gathered available online resources on the FSL alphabet. The frames from these videos are then extracted and used as image data. The authors were able to extract the FSL alphabet from the public instructional videos (Aguila Tutorial Videos, 2020; Asley Rara, 2018; Benilde SDEAS, 2013; Exequiel Chavez, 2021; FSL TUTOR, 2021; GVSP-VSO FSL dictionary, 2013; ISLVD By EJ, 2020). The data collected from this process is then utilized to train four CNN models which are the following: MobileNet, MobileNetV2, MobileNetV3, and EfficientNet. The training was done using Tensorflow (Abadi et al., 2016). The accuracy, precision, recall, f-score, and size of the model were then recorded as the model's metrics.

3.2 TinyML inference

After the training phase, the model is then converted into a TensorFlow Lite model. The optimization utilized for this is the 16x8 quantization mode. Here, the activation values are converted to 16-bit integers while the weights are converted to 8-bit integers. Tensorflow states that this improves accuracy while the achieved reduction in size is 3-4x of the original model. The models are then implemented on the Raspberry Pi 3 device. Here, a fraction of the testing dataset of size 435 is then utilized for inference on the embedded device. The inference time, accuracy, precision, recall, f-score, and size of the model. In inference, the images are not taken in real-time. Instead, both the image and the scri[t are run in Raspberry pi.

3.3 Metrics

The four main metrics utilized to evaluate the model performance are precision, recall, f-score, and accuracy. Accuracy is self-explanatory and thus does not require further explanation. Precision is the measure of how much can the classifier reject a false value from being positive given a category. Recall, on the other hand, is the ability of the model to find and categorized a positive sample correctly. Lastly, the F-score both scores with zero if one of the two scores is zero and 1 if both measure perfect scores. The formulae of the metric are expressed in equations 1 to 3.

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive} \qquad \text{(Eq. 1)}$$

$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \qquad \text{(Eq. 3)}$$

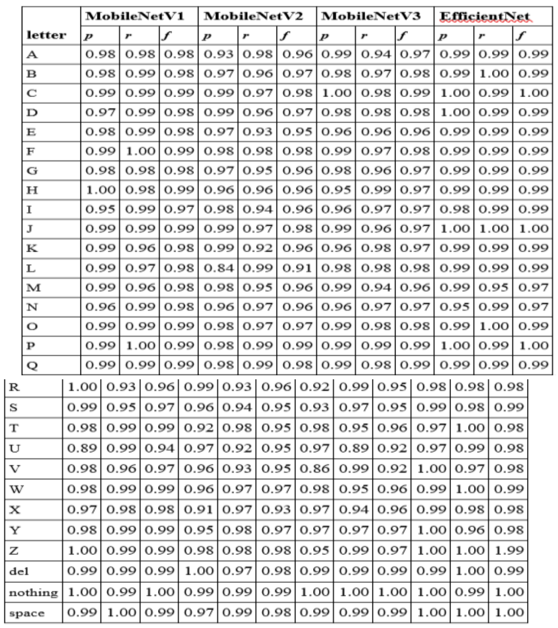
$$fscore = \frac{2 * Precision * Recall}{Precision + Recall} \qquad \text{(Eq. 4)}$$

4. RESULTS AND DISCUSSION

3.1 Training

A total of 204923 images were collected from the video and various sources The categories of the model are the 26 letters of the alphabet, two non-letters are deleted and space, and images that do not show any signs bumping the total number of categories to 29.

For the training and validation of the model, 70% of the database was used as training while 30% is used for testing. Four models were trained in this phase which are MobileNet, MobileNetV2, MobileNetV3, and EfficientNet. The batch size for training is 128 while the image size is 200x200px. The result of the metrics for the training model were shown in figure 1. Precision is p, Recall is r, while the f score is *f*



	MobileNetV1			**MobileNetV2**			**MobileNetV3**			**EfficientNet**		
letter	*p*	*r*	*f*	*p*	*r*	*f*	*p*	*r*	*f*	*p*	*r*	*f*
A	0.98	0.98	0.98	0.93	0.98	0.96	0.99	0.94	0.97	0.99	0.99	0.99
B	0.98	0.99	0.98	0.97	0.96	0.97	0.98	0.97	0.98	0.99	1.00	0.99
C	0.99	0.99	0.99	0.99	0.97	0.98	1.00	0.98	0.99	1.00	0.99	1.00
D	0.97	0.99	0.98	0.99	0.96	0.97	0.98	0.98	0.98	1.00	0.99	0.99
E	0.98	0.99	0.98	0.97	0.93	0.95	0.96	0.96	0.96	0.99	0.99	0.99
F	0.99	1.00	0.99	0.98	0.98	0.98	0.99	0.97	0.98	0.99	0.99	0.99
G	0.98	0.98	0.98	0.97	0.95	0.96	0.98	0.96	0.97	0.99	0.99	0.99
H	1.00	0.98	0.99	0.96	0.96	0.96	0.95	0.99	0.97	0.99	0.99	0.99
I	0.95	0.99	0.97	0.98	0.94	0.96	0.96	0.97	0.97	0.98	0.99	0.99
J	0.99	0.99	0.99	0.99	0.97	0.98	0.99	0.96	0.97	1.00	1.00	1.00
K	0.99	0.96	0.98	0.99	0.92	0.96	0.96	0.98	0.97	0.99	0.99	0.99
L	0.99	0.97	0.98	0.84	0.99	0.91	0.98	0.98	0.98	0.99	0.99	0.99
M	0.99	0.96	0.98	0.98	0.95	0.96	0.99	0.94	0.96	0.99	0.95	0.97
N	0.96	0.99	0.98	0.96	0.97	0.96	0.96	0.97	0.97	0.95	0.99	0.97
O	0.99	0.99	0.99	0.98	0.97	0.97	0.99	0.98	0.98	0.99	1.00	0.99
P	0.99	1.00	0.99	0.98	0.99	0.99	0.99	0.99	0.99	1.00	0.99	1.00
Q	0.99	0.99	0.99	0.98	0.99	0.98	0.99	0.98	0.99	0.99	0.99	0.99
R	1.00	0.93	0.96	0.99	0.93	0.96	0.92	0.99	0.95	0.98	0.98	0.98
S	0.99	0.95	0.97	0.96	0.94	0.95	0.93	0.97	0.95	0.99	0.98	0.99
T	0.98	0.99	0.99	0.92	0.98	0.95	0.98	0.95	0.96	0.97	1.00	0.98
U	0.89	0.99	0.94	0.97	0.92	0.95	0.97	0.89	0.92	0.97	0.99	0.98
V	0.98	0.96	0.97	0.96	0.93	0.95	0.86	0.99	0.92	1.00	0.97	0.98
W	0.98	0.99	0.99	0.96	0.97	0.97	0.98	0.95	0.96	0.99	1.00	0.99
X	0.97	0.98	0.98	0.91	0.97	0.93	0.97	0.94	0.96	0.99	0.98	0.98
Y	0.98	0.99	0.99	0.95	0.98	0.97	0.97	0.97	0.97	1.00	0.96	0.98
Z	1.00	0.99	0.99	0.98	0.98	0.98	0.95	0.99	0.97	1.00	1.00	1.99
del	0.99	0.99	0.99	1.00	0.97	0.98	0.99	0.99	0.99	0.99	1.00	0.99
nothing	1.00	0.99	1.00	0.99	0.99	0.99	1.00	1.00	1.00	1.00	0.99	1.00
space	0.99	1.00	0.99	0.97	0.99	0.98	0.99	0.99	0.99	1.00	1.00	1.00

Fig. 1. Model Training Metrics Results

As can be seen in the table above, the metrics for the model were high with some even achieving a score of 1.00. This high accuracy can also be seen in Table 1 wherein the accuracy of the four models is summarized

Table 1. Accuracy of Models

Model	Accuracy
MobileNet	0.9816
MobileNetV2	0.9637
MobileNetV3	0.9688
EfficientNet	0.9887

In Table 1, the best performing model is Efficient with MobileNet not being far behind. Version 2 and 3 of Mobilenet are in the 96% range.

3.2 TFLite Inference

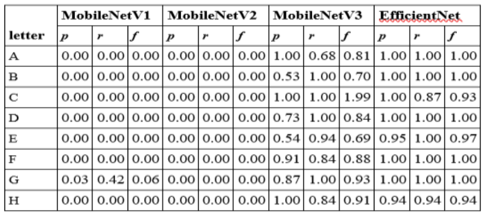
After recording the results of the training, the models were then converted into TFlite models. The conversion to TFlite utilized the 16x8 mode which can compress the model 3-4x times as well as make the inference more efficient. The size comparison pre and post-conversion is shown in Table 2.

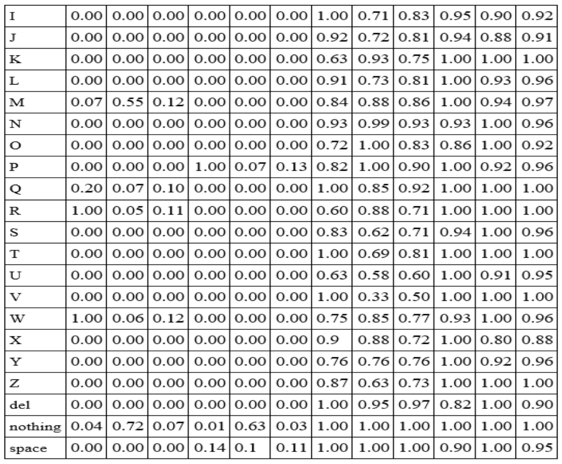
Table 2. Size of models

Model	Pre Conversion size (KB)
MobileNet	14540
MobileNetV2	11321
MobileNetV3	19350
EfficientNet	18941

Although Tensorflow claims a great reduction in size, the models were not reduced significantly as seen in the table above. These models were then implemented on a Raspberry Pi 3 device.

The models in the raspberry pi performed single language recognition inference on 435 images. The performance of these models can be seen in the metrics shown in fig 2





	MobileNetV1			MobileNetV2			MobileNetV3			EfficientNet		
letter	*p*	*r*	*f*	*p*	*r*	*f*	*p*	*r*	*f*	*p*	*r*	*f*
A	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.68	0.81	1.00	1.00	1.00
B	0.00	0.00	0.00	0.00	0.00	0.00	0.53	1.00	0.70	1.00	1.00	1.00
C	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00	1.99	1.00	0.87	0.93
D	0.00	0.00	0.00	0.00	0.00	0.00	0.73	1.00	0.84	1.00	1.00	1.00
E	0.00	0.00	0.00	0.00	0.00	0.00	0.54	0.94	0.69	0.95	1.00	0.97
F	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.84	0.88	1.00	1.00	1.00
G	0.03	0.42	0.06	0.00	0.00	0.00	0.87	1.00	0.93	1.00	1.00	1.00
H	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.84	0.91	0.94	0.94	0.94
I	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.71	0.83	0.95	0.90	0.92
J	0.00	0.00	0.00	0.00	0.00	0.00	0.92	0.72	0.81	0.94	0.88	0.91
K	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.93	0.75	1.00	1.00	1.00
L	0.00	0.00	0.00	0.00	0.00	0.00	0.91	0.73	0.81	1.00	0.93	0.96
M	0.07	0.55	0.12	0.00	0.00	0.00	0.84	0.88	0.86	1.00	0.94	0.97
N	0.00	0.00	0.00	0.00	0.00	0.00	0.93	0.99	0.93	0.93	1.00	0.96
O	0.00	0.00	0.00	0.00	0.00	0.00	0.72	1.00	0.83	0.86	1.00	0.92
P	0.00	0.00	0.00	1.00	0.07	0.13	0.82	1.00	0.90	1.00	0.92	0.96
Q	0.20	0.07	0.10	0.00	0.00	0.00	1.00	0.85	0.92	1.00	1.00	1.00
R	1.00	0.05	0.11	0.00	0.00	0.00	0.60	0.88	0.71	1.00	1.00	1.00
S	0.00	0.00	0.00	0.00	0.00	0.00	0.83	0.62	0.71	0.94	1.00	0.96
T	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.69	0.81	1.00	1.00	1.00
U	0.00	0.00	0.00	0.00	0.00	0.00	0.63	0.58	0.60	1.00	0.91	0.95
V	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.33	0.50	1.00	1.00	1.00
W	1.00	0.06	0.12	0.00	0.00	0.00	0.75	0.85	0.77	0.93	1.00	0.96
X	0.00	0.00	0.00	0.00	0.00	0.00	0.9	0.88	0.72	1.00	0.80	0.88
Y	0.00	0.00	0.00	0.00	0.00	0.00	0.76	0.76	0.76	1.00	0.92	0.96
Z	0.00	0.00	0.00	0.00	0.00	0.00	0.87	0.63	0.73	1.00	1.00	1.00
del	0.00	0.00	0.00	0.00	0.00	0.00	1.00	0.95	0.97	0.82	1.00	0.90
nothing	0.04	0.72	0.07	0.01	0.63	0.03	1.00	1.00	1.00	1.00	1.00	1.00
space	0.00	0.00	0.00	0.14	0.1	0.11	1.00	1.00	1.00	0.90	1.00	0.95

Fig. 2. Raspberry Pi TF-Lite Inference Metrics

Upon inspection of table 4., it can be said that only MobileNetV3 and EfficientNet performed well while the performance of MobileNetV2 and MobileNet is extremely subpar with most of the scores for the letters being 0.00. Fig.5 contains the confusion matrix of the inference. These figures can pinpoint the issues within the system

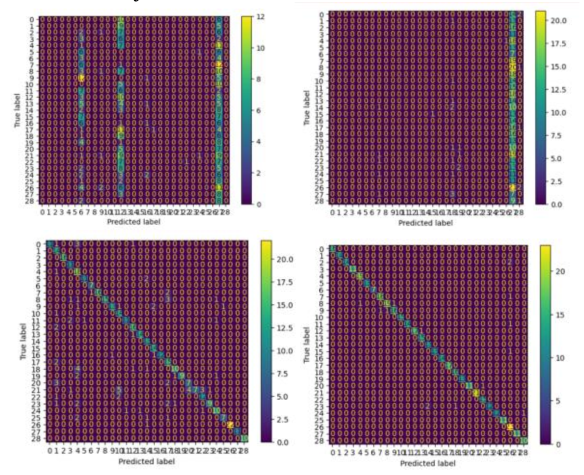


Fig. 3. confusion matrix of MobileNet V1, V2, and V3, and EfficientNet Respectively (

Fig. 5 shows that the model tends to put classify all of the signs into three categories which are 'G' (6), 'M' (12), and 'Nothing' (27). MobileNetV2 seems to put most of the categories on 'Nothing'. A possible explanation for this drastic accuracy dip and bias to certain categories is the quantization during conversion. It can be said that using vanilla MobileNet and MobilenetV2 do not convert well with quantization in TFlite due to it already being a portable model. A different story can be said on EfficientNet and MobileNetV3, wherein it performed better compared to the other two models with EfficientNet being more accurate. It can be inferred here that MobileNetV3 and EfficientNet did not significantly lower the model's performance. This can also be seen in Table 3 which summarizes the accuracy of the models with EfficientNet still being the dominant model. Aside from accuracy, inference time is also important due to the device being a resource constraint. Hence, the inference time is also shown below.

Table 3. TFlite models summary

Model	Accuracy	Inference time (s)
MobileNet	0.0517	224.69
MobileNetV2	0.0206	193.15
MobileNetV3	0.8160	183.69
EfficientNet	0.9655	550.13

Even though EfficientNet is more accurate, the inference time of MobileNetV3 is significantly less. Thus, MobileNetV3 should still be considered even with the accuracy dip if real-time performance is what is required of an implementation

4. CONCLUSIONS

Two TinML models running on Raspberry Pi 3 presented in this paper were able to perform Filipino Sign Language Recognition on image data. MobileNetV3 and EfficientNet scored 81.6% and 96.5% on accuracy with the precision, recall, and f-scores of some letters from the latter model achieving a score of 1.00. Although EfficientNet is more accurate than MobileNet, EfficientNet is significantly faster in inference which still makes it a compelling choice in embedded system application of FSL recognition. As for the two models—MobileNet and MobileNetV2—their accuracy is high before conversion, however, both models were unusable when converted into TFlite models. As for the size reduction, all of the models received a slight size reduction, however, it is not significant as all of these models are already portable. The authors of this study recommend extending this research to include other categories such as words, and to include other models. Focusing on implementing FSL recognition in real-life applications, such as mobile devices, is also advisable. Lastly, the use of other resource-scarce devices is also recommended for further studies.

5. ACKNOWLEDGMENTS

The authors of this paper would like to thank DOST for supporting the researcher via the Engineering Research Development for Technology or ERDT program.

6. REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2016). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*.

Aguila Tutorial Videos. (2020, September 21). *Learn the Filipino Sign Language Alphabet with Olivia Aguila*. https://www.youtube.com/watch?v=jVLJ63NkRPs

Al-Qurishi, M., Khalid, T., & Souissi, R. (2021). Deep Learning for Sign Language Recognition: Current Techniques, Benchmarks, and Open Issues. *IEEE Access*, *9*, 126917–126951. https://doi.org/10.1109/ACCESS.2021.3110912

Arredondo-Velázquez, M., Diaz-Carmona, J., Barranco-Gutiérrez, A.-I., & Torres-Huitzil, C. (2020). Review of prominent strategies for mapping CNNs onto embedded systems. *IEEE Latin America Transactions*, *18*(05), 971–982. https://doi.org/10.1109/TLA.2020.9082927

Asley Rara. (2018, November 5). *Alphabet in Filipino Sign Language (FSL)*. https://www.youtube.com/watch?v=wiagiGuxCO4

Benilde SDEAS. (2013, May 6). *FSL Alphabet*. https://www.youtube.com/watch?v=GD4d0HTp080

David, R., Duke, J., Jain, A., Reddi, V. J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., & Warden, P. (2021). *TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems*.

Exequiel Chavez. (2021, October 30). *FSL101: Learn the Filipino Sign Language (FSL) Alphabet*. https://www.youtube.com/watch?v=tiAdvY1ZiLw

Fischer, S. (2015). *Sign languages in their Historical Context* (pp. 442–465). https://doi.org/10.13140/2.1.2085.5683

FSL TUTOR. (2021, August 23). *The FSL Alphabet: Learn Filipino Sign Language Letter A-Z*. https://www.youtube.com/watch?v=bq6WzqvK76M

GVSP-VSO FSL dictionary. (2013, December 11). *Filipino Sign Language dictionary- Alphabet*. https://www.youtube.com/watch?v=TPpoRpgSZZo

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*.

ISLVD By EJ. (2020, December 8). *Basic Filipino Sign Language (FSL)—The Alphabet*. https://www.youtube.com/watch?v=2fX6wIGCnQA

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2017). *Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference*.

Jiang, S., Sun, B., Wang, L., Bai, Y., Li, K., & Fu, Y. (2021). *Skeleton Aware Multi-modal Sign Language Recognition*.

Montefalcon, M. D., Padilla, J. R., & Llabanes Rodriguez, R. (2021). Filipino Sign Language Recognition Using Deep Learning. *2021 5th International Conference on E-Society, E-Education and E-Technology*, 219–225. https://doi.org/10.1145/3485768.3485783

Newall, J. P., Martinez, N., Swanepoel, D. W., & McMahon, C. M. (2020). A National Survey of Hearing Loss in the Philippines. *Asia Pacific Journal of Public Health*, *32*(5), 235–241. https://doi.org/10.1177/1010539520937086

Notarte-Balanquit, L. (2021). *Insights from the First Filipino Sign Language (FSL) Summit (February 20-21) & the prospects for Filipino sign linguistics*.

Oliva, K. E., Ortaliz, L. L., Tobias, M. A., & Vea, L. (2018). Filipino Sign Language Recognition for Beginners using Kinect. *2018 IEEE 10th International Conference on Humanoid, Nanotechnology, Information Technology,Communication and Control, Environment and Management (HNICEM)*, 1–6. https://doi.org/10.1109/HNICEM.2018.8666346

Rastgoo, R., Kiani, K., & Escalera, S. (2021). Sign Language Recognition: A Deep Survey. *Expert Systems with Applications*, *164*, 113794. https://doi.org/10.1016/j.eswa.2020.113794

Sanchez-Iborra, R., & Skarmeta, A. F. (2020). TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities. *IEEE Circuits and Systems Magazine*, *20*(3), 4–18. https://doi.org/10.1109/MCAS.2020.3005467

Tan, M., & Le, Q. V. (2020). *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*.