Sign Language Recognition using Deep Convolutional Neural Networks

ST456 Deep Learning Project Report

Candidate 24738, Candidate 33747, Candidate 24299, Candidate 24262

ABSTRACT

Unlocking the language of gestures holds immense promise in breaking down communication barriers. This study aims to recognize Argentinian sign language gestures belonging to 10 different categories using advanced deep learning techniques, specifically via deep convolutional neural network (CNN) models. We evaluated several convolutional architectures in order to determine their applicability for this problem. Our approach involves training a model on frames extracted from videos. To avoid model overfitting, data augmentation, early stopping and dynamic learning rate reduction have been used. Various models have been compared, including VGG16 and VGG19, with and without data augmentation. Our best-proposed model achieved a high accuracy of 95.74%, highlighting its efficacy in recognizing Argentinian sign language gestures. Additionally, a heatmap overlay technique is utilized to elucidate the black-box neural network model.

Keywords: Sign Language, Sign Recognition, Convolutional Neural Network, VGG19, Video Processing

1 INTRODUCTION

Spoken language serves as the primary means of communication for a majority of the global population, facilitating widespread interaction. However, for a segment of society, spoken language proves inadequate in conveying their thoughts effectively. Sign language, a mode of expression largely characterized by visual hand gestures and movements, plays a pivotal role in facilitating communication for individuals with hearing or speaking impairments. Nonetheless, mastering sign language poses a formidable task, requiring dedicated training; this causes a communication bridge between those that know sign language and those that do not. It is also essential to recognize the inherent diversity within the distinct sign languages that exist across different regions and communities. Thus, even for individuals proficient in sign language, the ability to understand and interpret unfamiliar sign languages can pose significant challenges.

This study involves the recognition of a diverse set of 10 gestures in the Argentinian sign language. Our approach uses diverse methodologies ranging from video processing to feature extraction and the utilization of deep learning approaches in sign language recognition, each contributing to the advancement of this field. After conducting a comparative analysis of models with and without data augmentation, our best performing model is one that incorporates the VGG16 architecture along with seven supplementary layers. Achieving a final accuracy of 95.74% on the test dataset, our model exhibits promising performance especially with an emphasis on the effectiveness of data augmentation in improving sign language recognition.

Our research aims to contribute to the ongoing efforts in enhancing communication accessibility for individuals who rely on sign language as their primary mode of communication and to identify areas for further research in this field.

2 RELATED WORK

The field of sign language recognition using deep learning techniques has advanced significantly in recent years, paralleling developments in general image recognition and object detection technologies. This paper published by Sapijaszko and Mikhael (2018) [10], utlizes CNNs like AlexNet, VGGNet, ResNet and GoogleNet's Inception model.

In [11], the authors trained WideDenseNet and Prototypical Networks model for fewer data points and highlighted the importance of using data augmentation for better overall model performance. They also found that the Wide-DenseNet model performed very well for smaller samples, with an overall accuracy of 94% on RWTH dataset. Whereas, prototypical networks with data augmentation performed better for LSA16 dataset with around 99.26% accuracy. They have also used VGG16 on the LSA16 datatset which is a less complex but similar dataset to the LSA-64 dataset that we have used. Using VGG16 they get an accuracy of 95.92%, which is without using data augmentation.

Additional emphasis on data augmentation on fine-tuned CNN models has been placed by [9] this paper, where ResNet50 outperformed VGG16 based model with a test accuracy of 90% with data augmentation & 82% without data augmentation.

Furthermore, the VGG16 model was highlighted very highly in [7], where they were able to achieve an astonishing accuracy of 98% by using VGG16 with a Recurrent neural network model which in combination is called the YOLO (You Only Look Once) v5 mode, an advanced object detection algorithm.

Table 1: Similar approaches and their highest accuracy

Citation	Approach	Highest Accuracy
[11]	VGG16, w/o data aug	95.92%
[2]	Inception+SVM	93.67%
[2]	Inception	91.98%
[2]	Inception+NN	80.62%
[9]	ResNet50, VGG16 w data aug	90%
[9]	ResNet50, VGG16 w/o data aug	82%
[7]	VGG16 with RNN - YOLO model	98%

A brief explanation of some pre-trained models as mentioned in [10] is as follows:

- AlexNet: Introduced significant improvements in ImageNet classification challenges, pushing the boundaries of depth and complexity in neural architectures. Its application in sign language recognition could potentially improve the recognition of nuanced hand gestures over traditional methods due to its robust feature extraction capabilities.
- VGGNet (VGG-16 and VGG-19): Known for its deep architecture and the use of very small convolution filters, VGGNet provides an excellent framework for learning hierarchical features, which is crucial in sign language where the shapes and movements of hands are complex.
- Inception Models (GoogleNet, Inception-v3): These models introduce modules that optimize computational efficiency and accuracy by incorporating convolutions of varying sizes in one module. This approach is beneficial in sign language recognition for capturing spatial hierarchies in different scales, which can be critical in understanding subtle differences in sign language gestures.

We reviewed the accuracy scores in the literature of the models mentioned above and compared them. We came to realise that VGGNet had the highest accuracy scores for sign language data amongst other pre-trained convolutional neural net models.

A few methodologies that we adopted from the reviewed literature (as mentioned in [10]) are:

- **Feature Extraction**: As with the object recognition tasks described by Sapijaszko and Mikhael [10], sign language recognition relies heavily on effective feature extraction. The depth and complexity of models like VGG allow for detailed and nuanced understanding of the visual input, which is vital for distinguishing between similar signs.
- **Transfer Learning and Fine-tuning**: Similar to the approaches highlighted in the paper, transfer learning can significantly benefit sign language recognition. Pre-trained networks on large image datasets can be fine-tuned with smaller sign language datasets to improve learning efficiency and model performance.

3 CONTRIBUTIONS

Unlike many previous studies, which often rely solely on pre-trained architectures to perform sign language recognition, this paper introduces a novel approach by defining a VGG16 model with additional custom layers. This architecture allows for the extraction of more intricate features from the input images, thereby enhancing the overall performance of our sign recognition task.

Our research also explores the effectiveness of data augmentation techniques in improving the robustness and generalization capabilities of the model. By applying augmentations such as rotation, flipping, zooming, shifting, etc. to the training data, the model becomes more adept at handling variations in input images, leading to better classification accuracy. Such comprehensive augmentation techniques have not been thoroughly explored in previous literature especially with the dataset used in our work, marking a notable contribution to the field.

Additionally, adaptive learning rates are used to dynamically adjust the learning rate during training in order to obtain the globally optimum weights. This approach represents a notable advantage over traditional fixed learning rate methods, which may lead to suboptimal solutions.

4 DATASET USED

The dataset used in this research paper is called LSA64: A Dataset for Argentinian Sign Language, and is taken from the link mentioned in the references [8]. This dataset contains 3200 videos where 10 non-expert subjects executed 5 repetitions of 64 different types of signs. A total of 64 words have been shown in sign language which are shown in Figure 1. This paper works with 10 of these words to ensure efficient experimentation within the constraints of our available computational resources and also to reduce computational time.

ID	Name	H
01	Opaque	R
02	Red	R
03	Green	R
04	Yellow	R
05	Bright	R
06	Light-blue	R
07	Colors	R
08	Pink	R
09	Women	R
10	Enemy	R

Figure 1: Selected Words

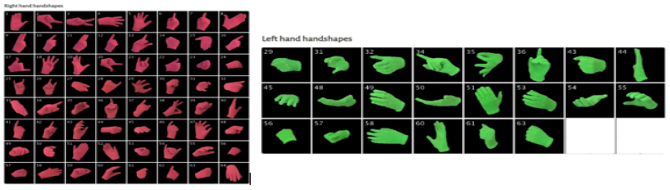


Figure 2: Sign language actions of right and left hand

4.1 Train-Test Split

The entire dataset of the videos has been divided into training and testing data. A split ratio of 0.8 was used to split the videos. The classes of each frame is extracted from the file name of its respective video, for example, the video file name "001_002_003" means that it is the video of the first "001" word which in our case (from 1) would be the word 'opaque'.

The train and test videos were then treated as explained below, for model fitting and testing purposes.

4.2 Processing Videos

To train the model for word recognition the following steps have been followed:

- **Extracting frames of images from the video**: Each frame has been extracted from the videos using the OpenCV library. This process involved capturing the individual image frames throughout the video duration.
- **Extracting hand segments from the frames**: Since the sign language in the videos dataset was made by wearing vibrant coloured gloves - pink in the right hand and green in the left hand - the hand segments have been extracted using colour boundaries of the pixels of each frame. Lower and upper limits to the colour boundaries have been set and since all the other background are more subdued colours of black and white, these colours were identified and hand segments were obtained. The before of the image and the after of the image is shown in 3.

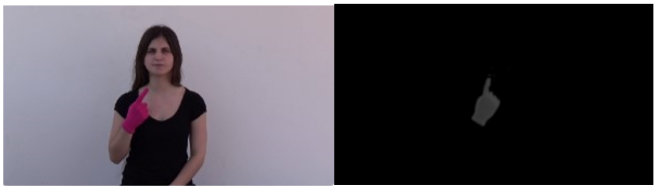


Figure 3: Video Frame along with extracted hand segment

- **Selecting middle 30% of the video frames**: Due to computational limitations, processing all frames of the videos together was not feasible. To construct a more time-efficient model, we selected the middle 30% of the frames of each video. This selection strategy ensures that the key moments of the sign gesture are included for model training while the transition is not captured. Due to this specific reason, in our study, training the Recurrent Neural Network (RNN) model on the 30% subset of frames did not yield the optimal accuracy (therefore, this has not been explored in detail in our research).

Each of these images has been resized with the following parameter set. This has been fixed for all the models to ensure consistency and replicability.

- `IMAGE_WIDTH = 64`
- `IMAGE_HEIGHT = 64`
- `IMAGE_CHANNELS = 3`
- `BATCH_SIZE = 128`
- `LABEL_CLASS = 10`
- `LEARNINGRATE = 1e-5`

5 MODEL ARCHITECTURE

To carry out the image recognition of the frames of the videos, deep Convolutional Neural Network models have been used. Our work builds and compares multiple updated versions of two pre-trained models which are the VGG19 and VGG16 models.

5.1 VGG19

VGG19 is an extension of the VGG16 architecture, featuring 19 layers, including 16 convolutional layers and 3 fully connected layers. VGG19 utilizes 3x3 filters with a stride of 1 and padding of 1 in all convolutional layers, followed by ReLU activation functions. VGG19 maintains the pattern of stacking convolutional layers with max-pooling layers for feature extraction.

The additional layers in VGG19 can allow for more complex feature representations compared to VGG16, potentially leading to improved performance on certain tasks. For this reason, we first attempt to use VGG19 as the base in our model architecture. It is also worth noting that while VGG19 is deeper than VGG16, it also comes with increased computational complexity and memory requirements.

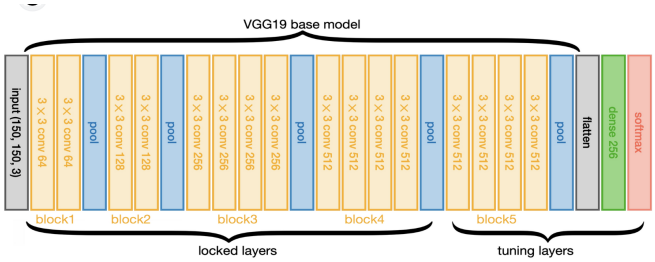


Figure 4: VGG19 Architecture [3]

We downloaded and used the weights for the pre-trained VGG19 model. Additionally, we have the following layers that we trained:

- A Flatten layer to flatten the input tensor into a one-dimensional array
- A Dense layer with 512 units and a ReLU activation function
- A Dropout layer with a dropout rate of 0.4 - a regularization technique we use to prevent overfitting
- A BatchNormalization layer to improve the training stability and speed
- A Dense layer with 512 units and a ReLU activation function
- A Dropout layer with a dropout rate of 0.3
- A BatchNormalization layer

Only in one instance during training, we changed the dropout rate of the first Dropout layer to check on its effect on the model performance; all other times, the layers were left unchanged.

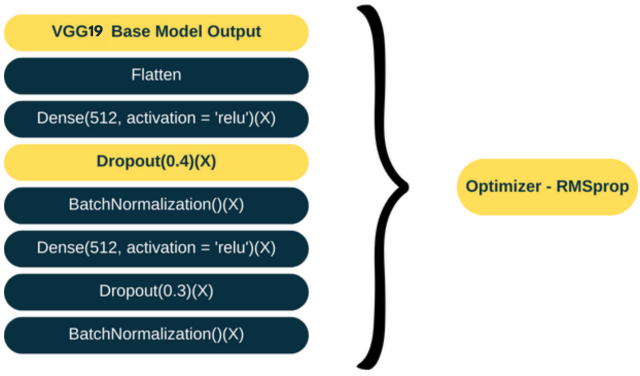


Figure 5: Additional Layers (the cells in yellow are modified in different models)

Finally we have a dense layer with 10 units (due to there being 10 distinct categories of signed gestures in our data), followed by a softmax activation function, serving as the output layer of the neural network.

5.2 VGG16

VGG16 is a CNN architecture consisting of 16 layers, including 13 convolutional layers and 3 fully connected layers. All convolutional layers in VGG16 use 3x3 filters with a stride of 1 and padding of 1, and they are followed by rectified linear unit (ReLU) activation functions. VGG16 follows a pattern of stacking multiple convolutional layers with max-pooling layers in between to progressively decrease the spatial dimensions of the input while increasing the depth. The final layers of VGG16 include three fully connected layers followed by a softmax activation function for multi-class classification. Despite its simplicity, VGG16 achieved state-of-the-art performance on the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2014.

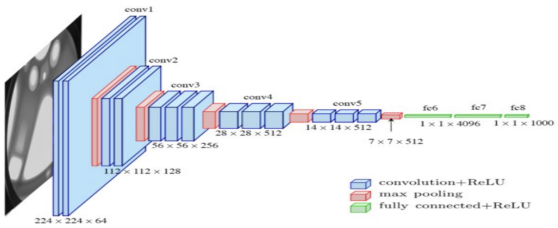


Figure 6: VGG16 Architecture [4]

We downloaded and used the weights for the pre-trained VGG16 model. Furthermore, we used the same additional layers as before for training.

6 TRAINING METHODS

The models have been trained for 10 epochs in total with the below-mentioned adjustments to their optimization function, the optimizer parameters, and other modifying conditions.

6.1 Optimizers Used

To optimize model convergence and enhance performance, two optimizers, namely RMSProp and Adam, were employed separately to check for changes in model performance and subsequently to find the best model. Usually, Stochastic Gradient Descent (SGD), Adam & RMSProp model optimizers are more commonly used in CNN model optimization [6]. RMSProp and Adam were selected in our paper based on their well-established performance in a wide range of deep learning tasks, including image classification and object detection.

6.1.1 Adam Optimizer. : Adam gradient considers the first and second moments of the gradient for optimization. This optimization algorithm, an extension of stochastic gradient descent, updates network weights during training. It blends features from both the "gradient descent with momentum" and the "RMSP" algorithms [1].

The equation for the gradient descent used under this is given in figure 7.

$$m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\delta L}{\delta w_t}\right] v_t = \beta_2 v_{t-1} + (1-\beta_2) * \left[\frac{\delta L}{\delta w_t}\right]^2$$

Figure 7: Adam Optimiser Equation

The image is referenced from [1].
Here,

- m_t: Aggregate of gradients at time t (current) (Initially, $m_t = 0$)
- m_{t-1}: Aggregate of gradients at time $t-1$ (previous)
- W_t: Weights at time t
- W_{t+1}: Weights at time $t+1$
- α_t: Learning rate at time t
- ∂L: Derivative of the loss function
- ∂W_t: Derivative of weights at time t
- β: Moving average parameter

6.1.2 RMSprop Optimizer. : RMSProp (Root Mean Square Propagation) addresses some of the limitations of traditional gradient descent methods by adapting the learning rates of individual parameters based on the magnitudes of recent gradients.

The update rule for parameters θ using RMSProp can be represented as follows:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{(1-\gamma)\, g_{t-1}^2 + \gamma\, g_t + \epsilon}} \cdot g_t$$

Figure 8: RMSProp Update Rule

The image is referenced from [6].
Here,

- θ_t is is the parameter vector at time step t.
- η is the learning rate.
- g_t is the gradient of the loss function with respect to θ at time step t.
- $(1-\gamma)g_{t-1}^2 + \gamma g_t$ is the exponentially decaying average of the squared gradients.
- ϵ is a small constant added to the denominator for numerical stability.

The intuition behind RMSProp is to adjust the learning rates for each parameter based on the magnitudes of their gradients. Parameters with larger gradients will have their learning rates scaled down more aggressively, while parameters with smaller gradients will have their learning rates scaled up, which can help speed up convergence and improve the stability of the training process.

Overall, RMSProp helps to adaptively adjust the learning rates for different parameters during training, which can be particularly useful when dealing with sparse data or in cases where the gradients of different parameters have vastly different scales.

The VGG19 model with additional layers (which we were training first) reached a higher accuracy of 95.71% on the test set when using the Adam optimizer as compared to a test accuracy of 95.66% with the RMSProp optimizer. This was why we used the Adam optimizer when training the VGG16 model with additional layers, as well.

6.2 Setting of Optimizer Parameters

Optimizing the parameters of the employed optimizers is also crucial to further enhance model performance. In our approach, careful consideration was given to configure these parameters to ensure optimal training outcomes.

We set LEARNINGRATE = 1e-5 as the initial value, after which a dynamic learning rate strategy was implemented during model training. This adaptive approach dynamically adjusts the learning rate during training by utilizing the ReduceLROnPlateau callback and monitoring the validation loss for a patience of 2 epochs. If the loss doesn't reduce for these 2 consecutive epochs then the learning rate is reduced, preventing the model from memorizing the training data excessively. By carefully setting the learning rate parameter, we enabled the model to overcome plateaus during training and achieve improved convergence towards the optimal solution.

The remaining parameters were set to the default values.

6.3 Methods used to mitigate model overfitting

Mitigating model overfitting is of paramount importance to ensure the generalizability and robustness of the trained model. To address this concern comprehensively, we employed two strategies: early stopping and data augmentation.

6.3.1 Early Stopping. This function has been added to the model training to stop the model training earlier if the validation loss stops decreasing for 5 consecutive epochs. This is done with the aim to optimize time utilization during training and enhance the model's efficiency by by curbing unnecessary training epochs. The optimal model weights are restored to ensure that the model is not over fitting. This way, post-training, the weights of that model are retained which yields the lowest validation set error.

6.3.2 Dropout. Dropout is a regularization technique used during training to prevent overfitting by randomly dropping out (setting to zero) a proportion of the neurons in the network during each training iteration. This helps to prevent co-adaptation of neurons and encourages the network to learn more robust features. By randomly deactivating neurons, dropout effectively simulates training multiple neural networks with different architectures, leading to improved generalization performance. During inference, dropout is typically turned off, and the full network is used for prediction.

6.3.3 Data Augmentation. Data augmentation aids in reducing model overfitting and improving the model's ability to generalize to unseen data. It serves as a powerful tool to expand the diversity of the training dataset by applying a variety of transformations to the input data [5]. Since sign language encompasses a wide range of gestures and signing styles with variations in hand shapes, movements and orientations, data augmentation techniques such as rotation, flipping and scaling can simulate these variations.

We defined data augmentation parameters for both the training and testing datasets using the data_gen_args1 and data_gen_args2 dictionaries. These parameters included specifications such as rotation range, width and height shift ranges, shear range, zoom range and flipping options. By augmenting the training data with variations in orientation, position and scale, we could provide the model with a more comprehensive understanding of the underlying patterns in the data.

7 NUMERICAL RESULTS

7.1 Goals of the Numerical Evaluations

The primary goal of the numerical evaluations is to evaluate the accuracy of the model in correctly recognizing and classifying sign language gestures depicted in our images. This involves measuring the overall classification accuracy of the model across different sign language gestures, as shown in Figure 9.

Another goal is to determine the model's ability to generalize to new images that it has not seen during training. This is crucial for evaluating the robustness and effectiveness of the model when it encounters previously unseen data. We evaluate the model's performance on a held-out test set containing unseen images to assess its ability to generalize beyond the training data; we aim to have a high accuracy of classification when it comes to unseen data.

7.2 Model Evaluation Metrics

Multiple evaluation metrics have been used to compare the performance of the models.

- **Accuracy**: Proportion of correct predictions -

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{1}$$

- **Loss**: Measure of how well the model performs on the model.
- **Precision**: Measure of the accuracy of the positive predictions made by a classification model.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

- **Recall**: Measure of the proportion of correctly predicted positive instances (true positives) out of all actual positive instances (true positives + false negatives)

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

- **Grad-CAM**: Simply put, Grad-CAM (which stands for Gradient-weighted Class Activation Mapping) looks at the pixels that were more important for the model when making a decision. It helps us visualise the hidden decisions made by the CNN, thus increasing interpretability of the model and aiding in the enhancement of its performance. We use a heatmap overlay technique to understand the contributing section of the image that impacts the final model prediction.

In our models, we aim to achieve high accuracy and low loss on the testing dataset. For the best performing model, it is also desirable to obtain high high precision and recall, indicating the model's accurate positive predictions and its comprehensive capture of the actual positive instances, respectively.

7.3 Comparison of Models

We have used two pre-trained models in our model architectures as the base: VGG19 and VGG16. Our first model with VGG19 and the additional layers is showed in 5.

Further changes are made to this and the model architectures are described in the table below (the final 3 cases are those where the models were trained and tested on augmented data).

Table 2: Model Architectures

Model No.	Architecture	Dropout rates	Optimizer	Act'n Func
1	VGG19 + 7 layers	(0.4, 0.3)	RMSprop	ReLu
2	VGG19 + 7 layers	(0.5, 0.3)	RMSprop	ReLu
3	VGG19 + 7 layers	(0.4, 0.3)	Adam	ReLu
4	VGG16 + 7 layers	(0.4, 0.3)	Adam	ReLu
1 ++	VGG19 + 7 layers	(0.4, 0.3)	RMSprop	ReLu
3 ++	VGG19 + 7 layers	(0.4, 0.3)	Adam	ReLu
4 ++	VGG16 + 7 layers	(0.4, 0.3)	Adam	ReLu

'++' : The model was trained and tested on augmented data.

The resulting accuracy and loss for validation and test sets are shown below.

Table 3: Test Accuracy of each Architecture

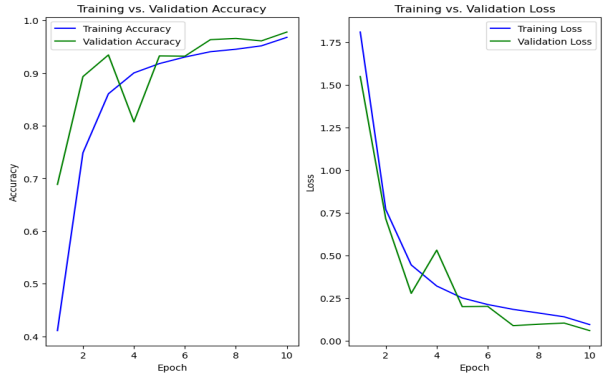
Model No.	Accuracy	Evaluation Loss
1	95.66%	0.1442
2	95.26%	0.1364
3	95.71%	0.1209
4	95.06%	0.1730
1 ++	94.19%	0.1883
2 ++	93.80%	0.1874
4 ++	95.74%	0.1460

++ : The model was trained and tested on augmented data.

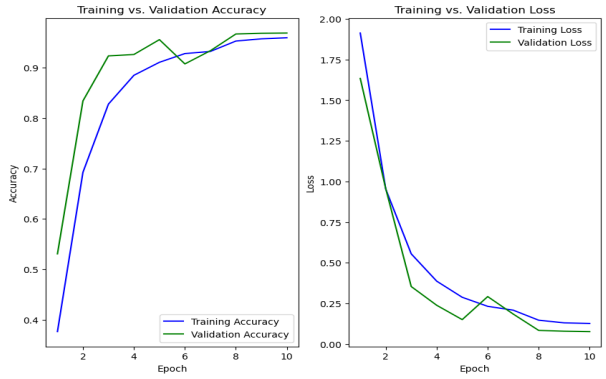
When examining the accuracy and loss trends (given in figure 9a) of the first model (denoted as Model 1 in table 2), we observe that during the fourth epoch, where there is a sudden drop in accuracy. Such anomalies in performance often indicate overfitting, a common issue in neural networks where the model memorizes the training data rather than learning to generalize from it, and can be often resolved by increasing dropout rates. Hence, the next model architecture (Model 2 in table 2) was tried with an increased dropout rate from 0.4 to 0.5. As evidenced by the resulting plot (9b), this adjustment mitigated the sudden drop in accuracy observed in the fourth epoch, suggesting that the dropout layer effectively regularized the model and reduced overfitting tendencies (although, model accuracy was slightly less in this case and so we did not implement the increased dropout rate in the remaining model architectures).

Furthermore, in the next model architecture (Model 3 in table 2), an alternative optimizer was implemented, i.e., the Adam optimizer. This optimizer is known for its adaptive learning rate capabilities and often yields smoother accuracy and loss function plots across epochs compared to traditional optimizers like SGD (Stochastic Gradient Descent). As can be seen from the accuracy and loss plots (9c), the utilization of the Adam optimizer in Model 3 did indeed result in more stable performance metrics over each epoch, indicating improved convergence and potentially enhancing the model's overall training dynamics.

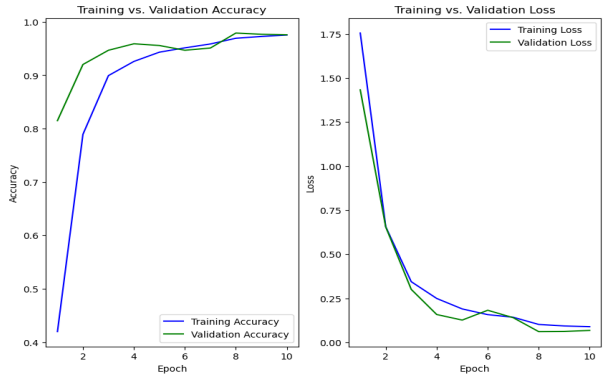
Thus, when we trained our model architecture with VGG16 as the base, we utilized the Adam optimizer.



(a) Model 1



(b) Model 2



(c) Model 3

Figure 9: Accuracy and Loss Plots

8 INTERPRETATION

8.1 Reflecting on the Model Architecture

We analyze the model architecture based on three parameters: Robustness, Interpretability, and Scalability.

- **Robustness**: With VGG16 and VGG19 as our base layers, the models can use transfer learning for better feature extraction. The additional fully connected layers have helped the model adapt its learned representations to the sign language classification task. They have acted as feature transformers mapping the high dimensional feature representation learned by VGG models to a space where they can be more easily classified by our model. The high accuracy with relatively fewer epochs signifies the greater efficiency of this design.
- **Interpretability**: The model follows a sequential design with two components (base layer and fully connected layers). This simplicity helps to better understand the flow of information through the model. Additionally, the use of Grad-CAM helps us visualise and explain the decisions made by the model, helping in the increase of interpretability.
- **Scalability**: Due to the sequential design, the model can be easily modified to handle increases in data volume and complexity, therefore making the model scalable.

8.2 Reflecting on the Model Performance

After comparing the results of our various model architectures that we implemented with and without data augmentation, our final model is the 4++ model from 2, which is the VGG16 model with 7 additional layers trained and tested on augmented data. The final accuracy that we get is 95.74% on the test data. As compared to other models we covered in our literature review, ours shows promising results provided that we applied data augmentation.

In all the models we saw earlier, for example in [9] where the accuracy is 90% with data augmentation and in [10] where it is 98% but without data augmentation - our model over-performs in comparison, in the case of using augmented data.

The following are the precision and recall values for each of our 10 individual classes, when using our best model for testing:

	precision	recall	f1-score	support
Opaque	0.94	0.99	0.97	357
Red	1.00	1.00	1.00	372
Green	1.00	1.00	1.00	344
Yellow	0.99	0.91	0.95	310
Bright	0.99	0.96	0.98	401
Light-Blue	0.91	1.00	0.95	455
Colors	0.90	0.83	0.87	343
Pink	0.91	0.99	0.95	286
Women	0.99	0.98	0.99	368
Enemy	0.93	0.88	0.91	310

Figure 10: Evaluation metrics for each class

We found it essential to include data augmentation since real-world sign language images may contain noise or slight distortions in hand signs that can affect the model's performance. Sign language gestures can also vary widely in terms of hand orientation, position, shape and background. Data augmentation techniques such as rotation, scaling and flipping can artificially increases the diversity of training data, allowing the model to learn to recognize gestures under different variations.

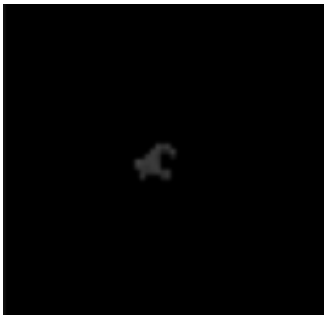
A probable reason that our model performs so well is that along with using the weights of the VGG16 model, we added an additional 7 non-convolutional layers. These additional layers, including 2 dropout layers for regularization and 2 batch normalization layers for faster convergence, can provide more flexibility for capturing higher-level features and relationships that may not be adequately captured by the convolutional layers alone.

Overall, our model performance is highly promising given the complexity of the data used, especially seeing that we were only able to train on 30% of the frames due to computational resource limitations, and for 10 epochs.

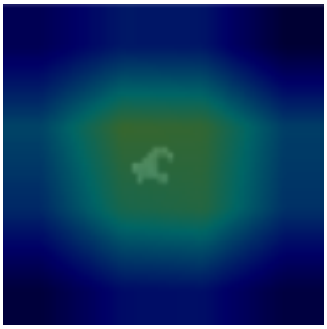
8.3 Sanity Checks using Explainable AI (GRAD-CAM)

To understand the model predictions better, we used the Gradient-weighted Class Activation Mapping (Grad-CAM). In our model, the additional dense layers make predictions by combining the features learned by the last convolutional layer of the VGG16 base model; therefore, we applied the Grad-CAM on the last convolutional layer in our model structure.

Figure 11: GRAD-CAM on last convolutional layer



(a) Original Frames



(b) Heatmap Image

The pair of images displayed above includes the original frame and its corresponding Grad-CAM heatmap overlaid on it. The heatmap reveals that the area around the sign is lighter in color, indicating that the model can successfully identify the area of the image that the hand sign lies in and assigns it greater importance. The orange-greenish radial pattern represents a general attention towards the center of the image. This is because in most of our input images the hand is generally in the centre of the image (due

to restricting our data to the middle 30% of the frames, which contain the actual signed gestures and no transition frames). Thus, the model has likely learned that generally, the central area of the image is to be focused on since the relevant information for prediction lies there.

We observe this radial green pattern in almost all our images in the last convolutional layer. This consistent pattern across multiple images suggests that it due to the common characteristic of the images being in the centre, that is the area our model focuses on.

9 CONCLUSION

9.1 Limitations

- **Using limited number of frames per video**: Using all the frames of the video would be an amazing way to enhance the model performance. Due to lack of computational resources, we were only able to accommodate middle 30% of the frames of a video/ Hence the CNN + RNN model performance for the sequential nature of the frames of videos data was only around 60%.
- **Subsetting the number of words in the dataset**: We subsetted only 10 words out of 64 words in the LSA64 dataset due to lack of computational resources.
- **Low number of epochs**: Running the model for a higher number of epochs would ensure better weights resulting in higher accuracy. With our models, even though we have run the model for 10 epochs, we were able to attain an accuracy of around 95.74%. However, running it for more epochs may result in better accuracy.

9.2 Future Work

- **Real-time Detection**: The model's performance can be evaluated real-time using programs like OpenCV and mediapipe. It is possible to develop techniques to improve the real-time performance of sign language detection models, enabling them to process video streams efficiently with minimal latency.
- **Cross-lingual Sign Language Recognition**: The model can be extended to support recognition of multiple sign languages, enabling communication across different linguistic communities. This could involve building multi-lingual datasets, or leveraging transfer learning techniques to transfer knowledge across languages. For example for the LSA64 dataset, since it is an Argentinian Sign Language dataset, a language translation model to Spanish can be created.
- **Running the code for all the frames of the Videos and using all words**: With higher computational capabilities a larger model can be trained using all the words in the LSA64 dataset and all the frames for every video.

10 BIBLIOGRAPHY

REFERENCES

[1] [n. d.]. https://www.shiksha.com/online-courses/articles/adam-optimizer-for-stochastic-gradient-descent/. ([n. d.]).

[2] Franco Ronchetti1 Laura Lanzarini1 Facundo Quiroga1, Ramiro Antonio1 and Alejandro Rosete. [n. d.]. A Study of Convolutional Architectures for Handshape Recognition applied to Sign Language https://sedici.unlp.edu.ar/bitstream/handle/10915/63481/Documento$_c$$ompleto.pdf-PDFA.pdf?sequence = 1isAllowed = y.([n.d.]).

[3] Jim Huang Srujay Korlakunta1 JenniferGrannen Alexander Robson Richard M.Allen Gaurav Chachra, Qingkai Kong. [n. d.]. https://www.nature.com/articles/s41598-022-12965-0. ([n. d.]).

[4] Yung-Tsun Tina Lee Kincho H. Law Max Ferguson, Ronay Ak. [n. d.]. https://www.researchgate.net/publication/322512435$_A$$utomatic_l$$ocalization_o$$f_c$$asting_D$$efects_w$$ith_c$

[5] Alhassan Mumuni and Fuseini Mumuni. 2022. Data augmentation: A comprehensive survey of modern approaches. *Array* 16 (2022), 100258. https://doi.org/10.1016/j.array.2022.100258

[6] Ramaprasad Poojary and Akul Pai. 2019. Comparative Study of Model Optimization Techniques in Fine-Tuned CNN Models. In *2019 International Conference on Electrical and Computing Technologies and Applications (ICECTA)*. 1–4. https://doi.org/10.1109/ICECTA48151.2019.8959681

[7] MOHAMMAD NADEEM 1 ROOBAEA ALROOBAEA 2 QAZI MOHAMMAD AREEB 1, MARYAM1 and FAISAL ANWER 1. [n. d.]. Helping Hearing-Impaired in Emergency Situations: A Deep Learning-Based Approach. ([n. d.]).

[8] Facundo Quiroga. [n. d.]. Dataset https://facundoq.github.io/datasets/lsa64/. ([n. d.]).

[9] Amit Kumar Mondal Ramaprasad Poojary, Roma Raina. [n. d.]. Effect of data-augmentation on fine-tuned CNN model performance. ([n. d.]).

[10] Genevieve Sapijaszko and Wasfy B. Mikhael. 2018. An Overview of Recent Convolutional Neural Network Algorithms for Image Recognition. In *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*. https://doi.org/10.1109/MWSCAS.2018.8623911

[11] 3 Franco Ronchetti1 Facundo Quiroga1 Waldo Hasperu´e1 2 Ulises Jeremias Cornejo Fandos1, Gaston Gustavo Rios1 and Laura Lanzarini1. [n. d.]. Recognizing Handshapes using Small Datasets. ([n. d.]).