Towards Sign Language Recognition Using Low-shot Learning

Gagandeep Kaur

Sarv Sanskriti School, pirthipalsingh138@gmail.com

Acknowledgement

Thank you for the guidance of Shreyank N. Gowda from University of Edinburgh in the development of this research paper.

***Abstract* – This research delves into enhancing sign language models through a combination of low-shot learning and Convolutional Neural Networks (CNNs). By examining the impact of varying sample sizes per iteration (k) on model accuracy across diverse datasets, we establish the effectiveness of these techniques. Initially employing a CNN model with low-shot learning produces promising results. Subsequent integration of ResNet-50 further amplifies accuracy, unveiling their potential in sign language detection and translation. The study underscores the consistent accuracy improvement with increased sample sizes, reaching 91.2% accuracy for ResNet-50 (110 samples) and 82.8% for CNN (1100 samples). This improvement saturates beyond a certain threshold, emphasizing the importance of dataset size and computational efficiency equilibrium for optimal training outcomes. These findings contribute valuable insights to robust sign language models, vital for efficient and accurate translation systems benefiting the deaf community.**

*Index Terms* – Convolutional Neural Network (CNN), Low-shot learning, and ResNet-50

**I. Introduction**

The realm of sign language translation and detection has garnered increasing attention in recent years due to its significance in facilitating effective communication for the deaf and hard of hearing community. The development of robust sign language models holds the potential to bridge the communication gap and empower individuals with hearing impairments [1]. Understanding the relevance and importance of this research is critical for justifying the efforts invested in addressing this challenging task. This study aims to explore the application of low-shot learning and deep learning models to enhance sign language models. By investigating the impact of different k-shot iterations on model accuracy across multiple datasets, we seek to identify the optimal approach for achieving improved performance in sign language translation and detection.

Deep learning has proven to be highly successful in various domains, including sign language detection, where it plays a pivotal role in facilitating seamless communication for the deaf community [2]. However, collecting and annotating vast amounts of sign language data can be labor-intensive and time-consuming [3]. Thus, it becomes essential to explore whether substantial datasets are indeed necessary to achieve optimal model performance.

This paper delves into the exploration of the impact of data quantity on the performance of deep learning models for classification, specifically within the context of sign language detection. The primary objective of this study is to understand how different data amounts influence the accuracy and efficiency of deep learning models and to ascertain the minimum data requirement to achieve comparable performance with that of using the full dataset. In doing so, we aim to address the fundamental question: "Do we truly need extensive volumes of data for effective Sign language detection?" By analyzing the classification performance across different data quantities, we seek to reveal valuable insights into the relationship between data volume and model accuracy. Understanding these connections will enable us to make better decisions regarding data collection and resource allocation for sign language detection systems. The findings of this research will contribute to the advancement of efficient and accurate sign language detection models, potentially revolutionizing communication for the deaf and hard of hearing community.

The study demonstrates that enlarging data samples typically leads to accuracy improvement in sign language detection, with optimal values around k = 1300 for smaller datasets. Both CNN and ResNet-50 models exhibit comparable trends of accuracy enhancement followed by saturation. However, resource-intensive ResNet-50 faces obstacles and overfitting risks. Effective data selection and efficient architectures, hence, are the key factors for achieving data-efficient model development, offering new insights for advancing sign language detection and translation methodologies.

This paper proceeds by setting the context for the research, discussing the significance of the study in light of sign language detection, and outlining the specific objectives and research questions to be addressed (Section I). Furthermore, it describes a literature review of the related work in this domain (Section II). Additionally, the background of this study (Section III) and research methodology have been provided for a better understanding of the methods used to obtain results (Section IV). Section V details the data results with a sub-section for graphs and tables, followed by Section VI, which highlights the results and discusses the key takeaways from this study. The paper concludes with Section VII and provides future directions.

**II. RELATED WORK**

**1. Introduction to Convolutional Neural Network (CNN) Architectures for Recognition:** The domain of deep learning has witnessed a remarkable emergence of Convolutional Neural Networks (CNNs), an intricate subset of neural networks with a distinctive aptitude for processing image-related tasks. Among the advancements within the realm of CNN architecture, one noteworthy progression is VGG16. This architecture, a refined iteration of AlexNet, made significant changes in image analysis due to its depth and adoption of diminutive convolutional filters. This strategic evolution was notably highlighted through its outperformance in the ImageNet Challenge. An equally important stride is exemplified by the ResNet architecture, characterized by its incorporation of residual mapping and a unique assembly of residual blocks. It resulted in improved optimization processes and increased accuracy, even when the network became more complex. These significant improvements clearly emphasize how important CNN architectures are for various tasks involving recognition and classification. This sets the basis for the current study, which focuses on refining and improving existing CNN models to perform exceptionally well in the complex area of sign language recognition. [4]

**2. Sign Language Recognition:** The vast and intricate realm of Sign Language Recognition (SLR) can be divided into two principal domains: the first being the recognition of individual signs, often termed "single sign recognition," and the second involving the identification of consecutive signs constituting sentences and complex gloss representations, known as "continuous sign recognition." The present research embarks on a pioneering approach focusing on the categorization of singular signs. To grapple with the multifaceted challenges posed by sign recognition, an array of strategies has been strategically employed. These strategies encompass the utilization of Recurrent Neural Networks (RNNs) are a leading technique for understanding complex hand alphabet symbols, like those made with fingers [5]. RNNs have shown good results, but we also need to recognize the past efforts of methods like the ones described in [6] and [7]. These methods used Fuzzy Min Max Neural Networks (FMMNN) to recognize individual symbols and words. Nevertheless, a discernible limitation lay in their inability to achieve real-time sign segmentation, attributed to their dependency on sign-level training paradigms. The field also includes Neural Networks (NNs), where [8] was the first to combine results from different layers, and [9] was a pioneer in using a Time Delay Neural Network (TDNN) to recognize American Sign Language (ASL) using 2D motion paths.

**3. Scene Classification Harnessing the Power of CNNs:** The intricacies inherent to scene-level classification for high-resolution images, coupled with the scarcity of training samples, have spurred researchers to forge innovative pathways. Among these, a seminal approach is presented within [10]. The crux of their methodology lies in the harnessing of a residual learning network, popularly known as ResNet, to extricate profound depth-related and low-level features from remote sensing images. These features, upon fusion, culminate in the creation of robust scene semantic feature structures. Afterward, the information was meticulously organized using a technique known as support vector machines (SVMs). The highest point of this effort is shown by an impressive accuracy rate of 95.71% when tested with the GF-2 dataset. This strongly emphasizes the power of CNN-based designs in understanding complicated scene categories.

**4. Sign Language Recognition Architectures:** A significant development in the field of sign recognition comes from the work of Sharma and Anand, as described in [11]. They've explored the application of deep learning techniques to recognize static Indian Sign Language (ISL). Their research strongly supports the effectiveness of the ResNet architecture in classifying both numerical and alphabetical ISL data. Their main finding is that carefully modifying CNN architectures and daringly creating networks from scratch often outperforms the results achieved by using pre-trained models like ResNet and Inception V3. This exploration serves as a compelling call to action, inspiring further research into improving existing pre-trained models. This ultimately leads to a significant improvement in their capabilities.

It's worth noting that in our study, we've also used a dataset related to Indian Sign Language, drawing a similarity to the work of Sharma and Anand.

**5. Unveiling Standard Chinese Natural Sign Language Recognition:** Working with the complex realm of recognizing Standard Chinese Natural Sign Language (CNSL), a groundbreaking effort unfolds in [12]. As outlined in this narrative, the intricate process of CNSL recognition is broken down into smaller, manageable tasks using a smart approach known as few-shot learning. This breakdown helps reduce costs and the time required.

The heart of this approach is the creation of the Cornerstone Network (CN) model. This model is made up of different parts, including an optional but effective feature extractor, an embedding network, and a semi-supervised clusterer that helps generate important elements. The CN model's greatest achievement is its exceptional success in achieving unmatched accuracy results with just five examples per class across established datasets. Remarkably, this is accomplished without relying on the crutch of a feature extractor. This achievement clearly outperforms other models that use different methods of measurement.

The model's effectiveness is further proven by its expansion of the CNSL database. This expansion smartly adds more samples from 80 to 139, incorporating additional signals that lead to a significant improvement in accuracy. This positive trend continues as the CN model is combined with a 1-D convolution feature extractor, resulting in real-time improvements in recognition accuracy. To summarize, this journey's account results in a significant impact on CNSL recognition. It demonstrates the potential of few-shot learning methods and the introduction of the Cornerstone Network model. The results reach their peak in improved accuracy across various datasets and real-life situations, highlighting the potential of innovative techniques.

**6. Sign Language Recognition Advancements:** Sign Language Recognition: Pioneering work by P. T. Krishnan and P. Balasubramanian [13] introduced a custom CNN architecture tailored for American Sign Language (ASL) alphabet translation. Their approach incorporated multiple convolutional and pooling layers, batch normalization, and dense layers. This architecture, trained from scratch, achieved an impressive accuracy of 82% on validation data. Furthermore, Ismail Hakki Yemenoglu et al. [14] conducted research on a CNN-based sign language recognition system. They employed the GoogleNet CNN architecture, adapted through transfer learning, to achieve a recognition accuracy of 91.02% on a dataset of character-level ASL signs. These foundational works emphasize the significance of tailored CNN architectures in advancing the field of sign language recognition.

**III. BACKGROUND**

**CNN (Convolutional Neural Network):** CNN, referred to as Convolutional Neural Network [15], belongs to the category of neural networks designed to leverage the spatial arrangement of input data. These CNN models exhibit a standard architecture that comprises interleaved convolutional and pooling layers, often placing each pooling layer following a convolutional one. Concluding these layers are a few fully-connected layers, culminating in a softmax classifier, illustrated in Figure 1. The training process for CNNs generally involves utilizing backpropagation through Stochastic Gradient Descent (SGD), which fine-tunes the weights and biases to minimize a specified loss function. This optimization aims to closely align arbitrary inputs with the desired outputs.

Figure 1 illustrates the arrangement of a CNN, encompassing convolutional, pooling, and fully-connected layers. The convolutional layer is composed of a collection of adaptable kernels or filters that extract localized features from the input data. Each kernel calculates a distinctive feature map, with units of these maps exclusively linking to limited portions of the input, known as receptive fields. A new feature map emerges by sliding a filter across the input and computing a dot product (akin to convolution), succeeded by a non-linear activation function that introduces model non-linearity. Notably, all units share the same weights (filters) across each feature map. Weight sharing offers the advantage of fewer parameters and the ability to detect identical features regardless of their position within the input data [16].

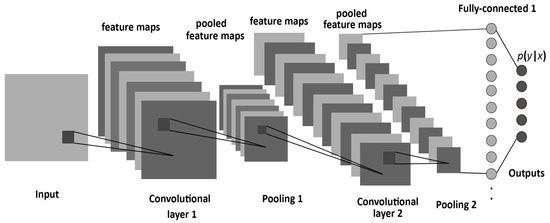


FIGURE 1 [21]

the structure of a CNN, consisting of convolutional, pooling, and fully-connected layers

Various non-linear activation functions, like sigmoid, tanh, and ReLU, are available. However, ReLU [f(x) = max (0, x)] is favored due to its expeditious training relative to others [17, 18]. The size of the output feature map relies on the filter dimensions and the stride. Given an input image convolved with a filter of size (H × H), employing a filter of size (F × F) and stride (S), the resulting output size (W × W) is governed by the equation:

Pooling is a method that reduces the resolution of earlier feature maps, like zooming out. This helps the system ignore small changes or distortions. Pooling divides the inputs into separate areas of a certain size (usually represented as R × R) and produces one output for each area [19]. Pooling can take the form of maximum or average-based operations [20]. For an input of dimensions (W × W) presented to a pooling layer, the resulting output size (P) can be determined by:

The upper sections of CNNs have fully-connected layers, similar to the ones in regular neural networks. These layers aim to capture overall features from the inputs. The units in these layers connect with all the hidden units in the previous layer. The final layer works as a softmax classifier, figuring out the likelihood of each class label among the given k classes.

Formula of Softmax function (Equation 1):

=

=

=

**ResNet-50 (Residual Network with 50 layers):** In the domain of deep learning, Convolutional Neural Networks (CNNs) have played a pivotal role in achieving cutting-edge performance across a range of computer vision tasks. Nonetheless, as these networks grew more complex, they encountered a well-known obstacle referred to as the vanishing gradient problem [22]. It posed challenges to training networks having a substantial number of layers. The inception of the ResNet-50 architecture marked a significant breakthrough in tackling this predicament.

ResNet-50, denoting "Residual Network with 50 layers," was introduced by Kaiming He et al. in their paper titled "Deep Residual Learning for Image Recognition" [23], presented at CVPR 2016. The foundational innovation of ResNet-50 revolves around the concept of residual learning. This innovation entails incorporating skip connections that facilitate a more direct flow of information within the network. These connections empower the network to learn residual mappings, capturing the discrepancy between the current output and the desired output. This mechanism enables gradients to flow more effectively during backpropagation, effectively countering the degradation of performance that often affects deep learning models due to vanishing gradients. As a result, ResNet-50 adeptly addresses the vanishing gradient problem, allowing for the training of remarkably deep networks with improved convergence and precision.

The architecture of ResNet-50 is defined by its modular structure, comprised of consecutive building blocks referred to as residual blocks. Each residual block contains a series of convolutional layers, accompanied by batch normalization and non-linear activation functions, all within a skip connection [24]. This design choice not only facilitates efficient learning but also simplifies the training of networks with a large number of layers. This adjustment introduces a dual-path information flow: the original route, which directly traverses the block, and the shortcut path, bypassing one or more layers. This architectural innovation effectively addresses challenges such as vanishing gradients, allowing for successful training of exceptionally deep networks and promoting improved model performance.

As a result, ResNet-50 has evolved into a cornerstone in diverse computer vision applications, encompassing tasks such as image classification, object detection, and segmentation. This underscores its pivotal role in propelling the progress of deep learning models for the analysis of visual data.

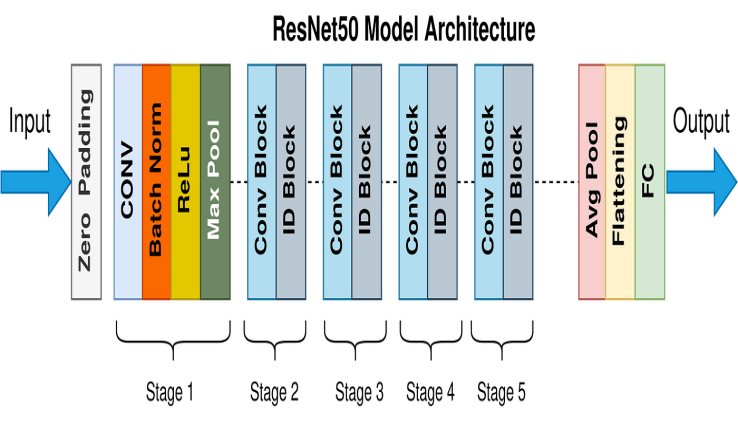


FIGURE 2 [25]

The structure of a ResNet-50 architecture

**IV. METHODOLOGY**

In this section, we elucidate the rigorous methodology employed to conduct our investigation into the research question. A well-structured methodology is essential to guide our study's progression, ensuring precision in data collection, analysis, and interpretation. By delineating the systematic procedures adopted, we aim to provide readers with a comprehensive understanding of how we approached our research objectives.

**Data Collection***:* For this study, we conducted data collection from publicly available Kaggle datasets, with a specific focus on three primary sources: 'Sign language MNIST,' and 'Indian sign language,’ and ‘Sign language for numbers.’ These datasets along with the MNIST dataset were chosen to ensure a diverse representation of sign language gestures and numerical signs (0 to 9), enabling a comprehensive evaluation of the sign language detection model's performance.

The 'Sign language MNIST' [26] dataset served as the cornerstone for model training and experimentation. It comprises grayscale images of sign language gestures. A key strength of this dataset lies in the active participation of individuals with diverse sign language backgrounds and proficiency levels, who voluntarily contributed to the collection of these hand gestures. To ensure sufficient number of training samples per class (various signs and gestures), participants performed multiple instances of each sign gesture, enriching the dataset's quality and diversity. To assess the model's ability to recognize sign gestures from different sign languages, we introduced the 'Indian sign language' dataset [27]. This collection incorporates a variety of sign gestures commonly used in Indian Sign Language. Fluent participants from India actively recorded themselves performing various gestures, enhancing the dataset's diversity and cross-lingual representation. Furthermore, to evaluate the model's performance specifically for numerical signs (0 to 9), we incorporated the 'Sign language for numbers' dataset [28]. This dataset features sign gestures representing the digits in various sign languages from different regions, allowing for a comprehensive assessment of the model's generalization capability across numerical signs.

The inclusion of the MNIST, together with the 'Sign language MNIST' dataset and 'Indian sign language' datasets, ensured a well-rounded and inclusive approach to data collection and these datasets enabled a thorough evaluation of the sign language detection models. (*In this paper, we refer to MNIST dataset as Dataset 1, Sign Language MNIST’ dataset as Dataset 2‘Indian sign language’ dataset as Dataset 3 and ‘Sign language for numbers’ as Dataset 4)*

**Data****Preprocessing:** Before model training, the datasets were preprocessed in Google Colab. The data preprocessing steps included sorting the datasets according to individual numbers and letters, uploading the datasets to Google Drive and mounting Google Drive in the Colab environment for easy access during training. These data preprocessing steps ensured that the models received clean and suitable input, laying the foundation for a successful and meaningful analysis of the impact of data quantity and network architecture on the performance of the sign language detection models.

**Baseline CNN Model:** The research began by implementing a baseline Convolutional Neural Network (CNN) model for sign language detection. The CNN architecture was chosen due to its proficiency in image-based tasks [29]. The baseline CNN model consisted of two convolutional layers, followed by two max-pooling layers to extract essential features from the sign language images. The convolutional layers were designed to identify patterns and relevant local features within the images, while the max-pooling layers downsampled the feature maps to retain the most important information. Subsequently, the extracted features were flattened and passed through fully connected layers to perform classification.

Further, we have utilized two essential activation functions to code the models. Firstly, the Rectified Linear Unit (ReLU) has been used in the deep learning models. It introduces non-linearity by returning the input value if it is non-negative and zero otherwise [30]. This property enabled the network to learn complex relationships between inputs and outputs effectively. ReLU is computationally efficient and helped to mitigate the vanishing gradient problem during backpropagation. Secondly, the Log Softmax activation function (Equation 1) is employed for multi-class classification tasks. It transforms the output scores of the model into log-probabilities, making it suitable for predicting one of several classes. By taking the exponent of each input element, computing the logarithm, and then normalizing the result, Log Softmax [31] produced a probability distribution over classes. Furthermore, we used the CrossEntropyLoss function to calculate the loss during training. This loss function internally combines Log Softmax and Negative Log Likelihood Loss (NLL Loss) to compute the cross-entropy loss. The CrossEntropyLoss (Equation 2) measures the dissimilarity between the predicted probabilities and the true class labels. By including Softmax internally, CrossEntropyLoss ensured stable and accurate loss computation for multi-class classification problems, making it suitable for the training of classification models.

Formula of CrossEntropyLoss function (Equation 2):

*n = number of classes*

**Collecting Data:** This step involved the acquisition of data, an important process in the study. Two key aspects were addressed, namely, varying k and changing datasets. To comprehensively assess the influence of data quantity, the experiment systematically adjusted the number of samples per class (k). The training process was conducted across various k values, meticulously recording the corresponding accuracy scores. This important stage is the main focus of the study, aiming to determine the smallest amount of data required to achieve a performance similar to that of the entire dataset.

Along with using different k values, the study also encompassed the vital step of transitioning datasets. We modified our baseline model to change the dataset (Dataset 1, Dataset 2, Dataset 3 or Dataset 4) with an aim to determine the optimal k value where accuracy approached its peak for different datasets. This approach, of both modifying k values and datasets, was pivotal in identifying the optimal conditions for achieving near-maximal accuracy.

**Modifying the Neural Network Architecture to ResNet-50:** Further, the network architecture is updated from CNN to ResNet-50 in the baseline model. The model modification process initiates by establishing the necessary environment, importing essential libraries like `torch` and `torchvision`, along with modules dedicated to network design, data handling, and optimization. Effective model training begins with meticulous data preprocessing. The transformation pipeline, constructed using `transforms.Compose`, orchestrates the transformation of images. Initial resizing of images to dimensions of 224x224 pixels aligns with the ResNet-50 architecture's anticipated input size. Subsequently, the images undergo conversion into tensors, fundamental structures employed for computations in PyTorch. Next, the model's foundation is laid by loading the MNIST dataset for training via a `DataLoader`. The dataloader streamlines the training process through batching and data shuffling.

The model's core is the ResNet-50 architecture (Section III), a robust deep neural network [23]. The model is trained through a dedicated loop that iterates the training dataset over a set number of epochs. In each epoch, the model evolves based on deviations between predicted outcomes and actual labels. This discrepancy is evaluated using the Cross-Entropy loss function (Equation 2). The optimization process involves gradient computation, backpropagation, and leveraging the Adam optimizer to iteratively update the model's parameters. Upon successful training, the model's efficacy is gauged using the Dataset 2. During testing, the model's predictions are generated for the test data, subsequently compared to the actual labels. Precision is determined by dividing the count of accurately predicted labels by the total number of samples in the test set. Furthermore, utilizing a CPU for training, the ResNet-50 model is optimized to efficiently learn and categorize patterns in the input data, thus enabling accurate and expedited image classification.

While this model serves as a foundational structure for classification tasks, the utilization of an intricate architecture like ResNet-50 may exceed the requirements of the comparatively straightforward Dataset 1, 2 and 4. Consideration for simpler architectures like convolutional neural networks (CNNs) could potentially yield analogous outcomes with fewer parameters, mitigating the risk of overfitting.

**Statistical Analysis:** By systematically altering the value of k, the study acquired a comprehensive perspective on the relationship between data quantity and model performance. For each dataset, a series of k values were tested, and the corresponding accuracy scores were recorded. These values were then plotted on the graphs, with k along the x-axis and accuracy along the y-axis.

Generally, the accuracy exhibited an upward trend as k increased, reflecting the positive impact of a larger training set. However, the graphs also exhibited diminishing returns, where accuracy plateaus or experiences only marginal improvements beyond a certain k value. This observation could signify that the model's capacity has been saturated, and increasing k further might not lead to significant accuracy gains. Moreover, comparing the k-accuracy graphs across different datasets offered a unique perspective on the inherent characteristics of each dataset. Variations in the steepness of accuracy curves or the saturation points could indicate dataset-specific nuances. This information proved invaluable in tailoring the data preprocessing steps and model architecture to suit each dataset's unique attributes. In essence, plotting and analyzing k-accuracy graphs for different datasets allowed for a nuanced understanding of the relationship between data quantity and model performance. This insight guided the determination of optimal k values for achieving near-maximal accuracy, while also highlighting how dataset characteristics influenced this relationship. This step ultimately facilitated the fine-tuning of the model's training process for each dataset, optimizing its performance for real-world applications.

The methodology outlined above allowed for a systematic exploration of the impact of data quantity and network architecture on sign language detection accuracy. By utilizing Kaggle datasets and Google Colab for coding and experimentation, the study ensured the availability of publicly accessible resources and facilitated reproducibility. The resulting graphs and analyses shed light on the potential of low-shot learning, CNN and ResNet-50 in enhancing sign language detection systems.

**V. RESULTS**

The study's outcomes unveil a multifaceted understanding of the intricate interplay between data quantity and model efficacy within the context of sign language detection. We plotted k-accuracy graphs and conducted a thorough analysis of their implications. This step was crucial for comprehending how varying the number of samples per class (k) influenced the accuracy of the model’s predictions. Distinct graphs were plotted for each dataset under consideration.

Across different datasets, the number of samples is generally positively correlated with the model accuracy. However, this relationship is neither linear nor universal. The dataset-specific nature of this correlation is exemplified by critical points around k = 1300 (Figure 3), where the accuracy gains largest values for smaller datasets (Dataset 1, Dataset 2, and Dataset 4 of sizes 47 MB, 66 MB and 50 MB respectively). Intriguingly, this saturation point occurred at a much higher k value (Figure 6) for the larger dataset (Dataset 3), which boasted a size of 270 MB. The critical points we noticed show how dataset size and the points where improvements slow down are connected. This highlights the delicate balance between how much data we have and how much it helps improve the model. Notably, this phenomenon underscores the significance of dataset complexity and its influence on the saturation point, suggesting that smaller datasets exhibit an earlier limit in accuracy improvements. This observation holds particularly true for datasets with more limited sizes, reinforcing the notion that judicious data selection and efficient learning techniques can yield substantial results even within constrained resources.

The study states the nuanced and dataset-dependent nature of the relationship between data quantity and model performance. While increasing data samples often leads to accuracy improvement, the existence of critical points and saturation occurrences highlights the importance of a context-sensitive approach. These insights, along with the efficacy of low-shot learning techniques, challenge conventional assumptions that large amount of data leads to better accuracy, suggesting that even with limited data volumes, substantial advancements can be made in the realm of sign language detection.

Analyzing the data obtained from the baseline model using the ResNet-50 architecture for Dataset 2 (Figure 5) reveals an interesting pattern. As we increased the value of k, there was a significant improvement in accuracy up to a certain point, but afterward, the accuracy gains seemed to stabilize, fluctuating between 0.8 and 0.9. This finding suggests that the ResNet-50 model, with its deeper architecture and advanced features, initially benefited from increased data but eventually reached a saturation point in terms of accuracy improvement.

Now, let's compare this data with the results obtained from the CNN model for Dataset 2 (Figure 4). In the CNN model's results, there was also a notable improvement in accuracy as the value of k was increased, and a similar fluctuation was observed once the accuracy reached around 0.8. This suggests that both the CNN and ResNet-50 models exhibited a similar behavior in terms of accuracy improvement and saturation when faced with increasing amounts of data. The comparison indicates that while the two different architectures led to slightly varying accuracy values, the general trends of improvement followed by saturation remained consistent.

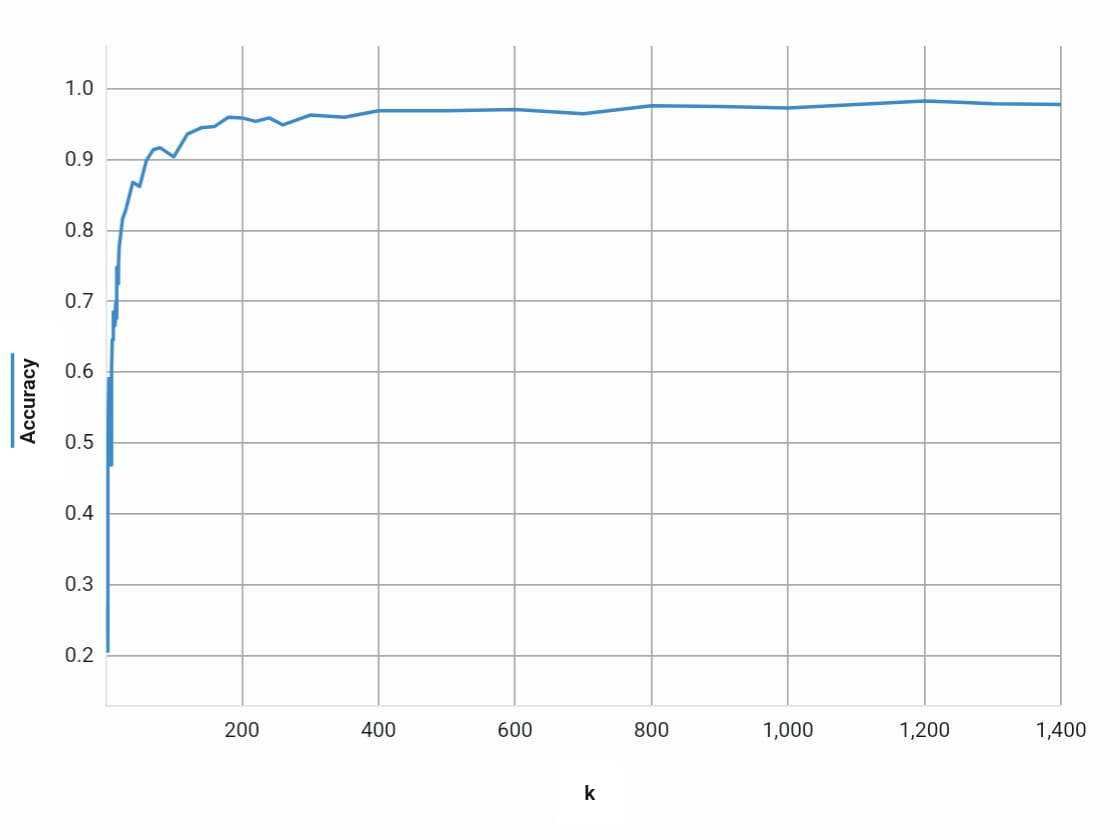
While assessing the performance of the ResNet-50 architecture in comparison to the CNN model, we encountered challenges in acquiring data for the ResNet-50 architecture using Dataset 1, 3 and 4. The utilization of the ResNet-50 architecture posed significant resource demands. Our attempts to execute the codes were hindered by rapid RAM consumption, leading to execution delays and infeasibility for these datasets. This constraint sheds light on the resource-intensive nature of deep architectures like ResNet-50 [32] and underscores the importance of considering computational limitations in model selection. Moreover, the challenges encountered with the ResNet-50 architecture highlight the potential risk of overfitting, a phenomenon where a model becomes too specialized in learning from its training data to the detriment of its generalization abilities. The resource constraints and execution delays encountered during training could aggravate the risk of overfitting, especially when working with smaller datasets. These limitations emphasize the delicate balance between model complexity, dataset size, and computational resources [33].

**Figures, Tables and Equations**

|  |  |  |
| --- | --- | --- |
|  | k | Accuracy |
|  | 100  200  300  400  500  600  700  800  900  1000  1100  1200  1300 | 0.904  0.959  0.963  0.969  0.969  0.971  0.965  0.976  0.975  0.973  0.978  0.983  0.979 |

TABLE I

k and accuracy values for dataset 1 by using cnn architecture: achieved accuracies range from 90.4% to 98.3% as samples increase from 100 to 1300, accuracy steadily improves, peaking at 97.9% for 1200 samples.

FIGURE 3

Accuracy-k graph for dataset 1 using CNN architecture (robust and progressive learning demonstrated

|  |  |  |
| --- | --- | --- |
|  | k | Accuracy |
|  | 50  100  200  300  400  500  600  700  800  900  1000  1100  1200 | 0.561  0.698  0.793  0.781  0.852  0.903  0.905  0.866  0.866  0.910  0.920  0.899  0.841 |

TABLE II

k and accuracy values for dataset 2 obtained using CNN architecture Despite a small dataset, the CNN model showcases remarkable accuracy. Achieving 79.3% accuracy at k=200 and 92.0% at k=1000



FIGURE 4

accuracy-k graph for dataset 2 obtained using CNN architecture (architecture highlights the model’s ability to effectively leverage available data)

|  |  |  |
| --- | --- | --- |
|  | k | Accuracy |
|  | 10  20  30  40  50  60  70  80  90  100  110  120 | 0.531  0.720  0.734  0.816  0.846  0.879  0.846  0.882  0.922  0.875  0.912  0.891 |

TABLE III

k and Accuracy values for dataset 2 obtained using ResNet-50 architecture. accuracy reaches 92.2% accuracy with 90 samples. Performance plateaus beyond 90 samples and we get diminishing returns in accuracy gains.

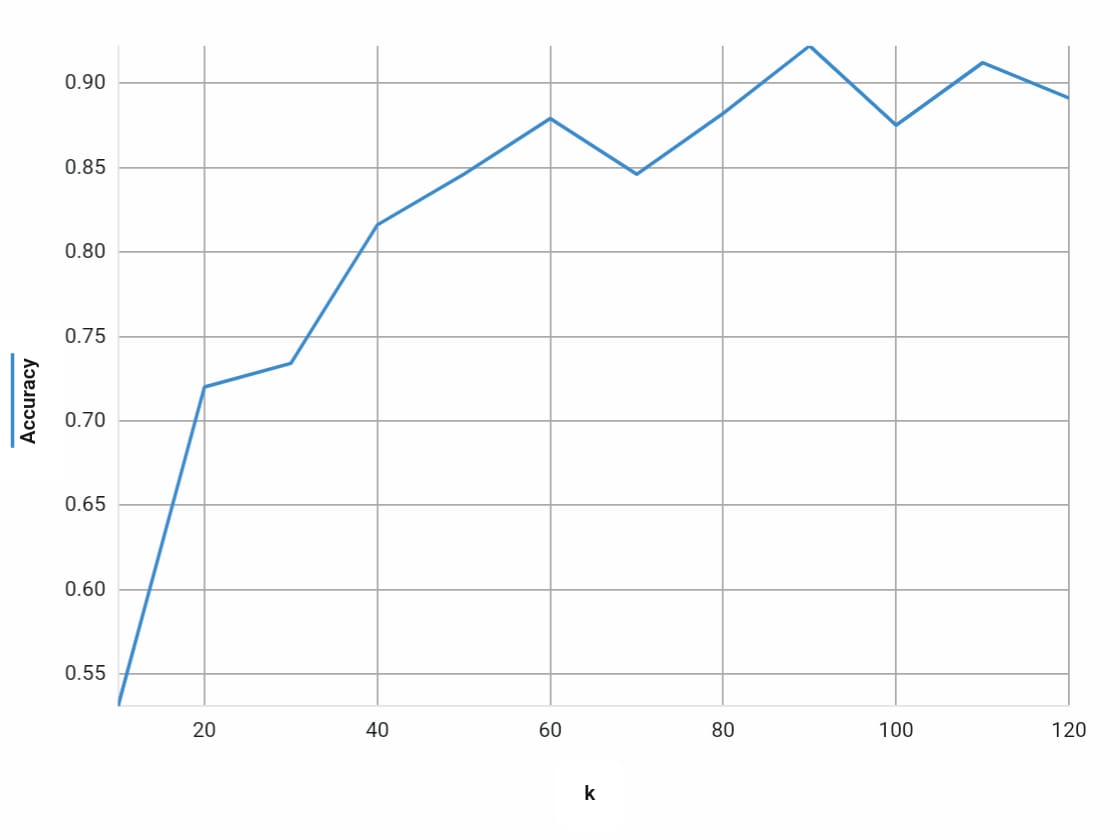


FIGURE 5

accuracy-k graph for dataset 2 obtained using ResNet-50 architecture (progressive learning demonstrated)

|  |  |  |
| --- | --- | --- |
|  | k | Accuracy |
|  | 100  200  300  400  500  600  700  800  900  1000  2000  8000 | 0.930  0.950  0.962  0.968  0.971  0.974  0.973  0.972  0.972  0.977  0.980  0.991 |

TABLE IV

k and Accuracy values for dataset 3: The CNN model demonstrates remarkable accuracy enhancement as samples per iteration increase, peaking at 98.9% accuracy with 3000 samples, revealing the model's robust learning capacity.

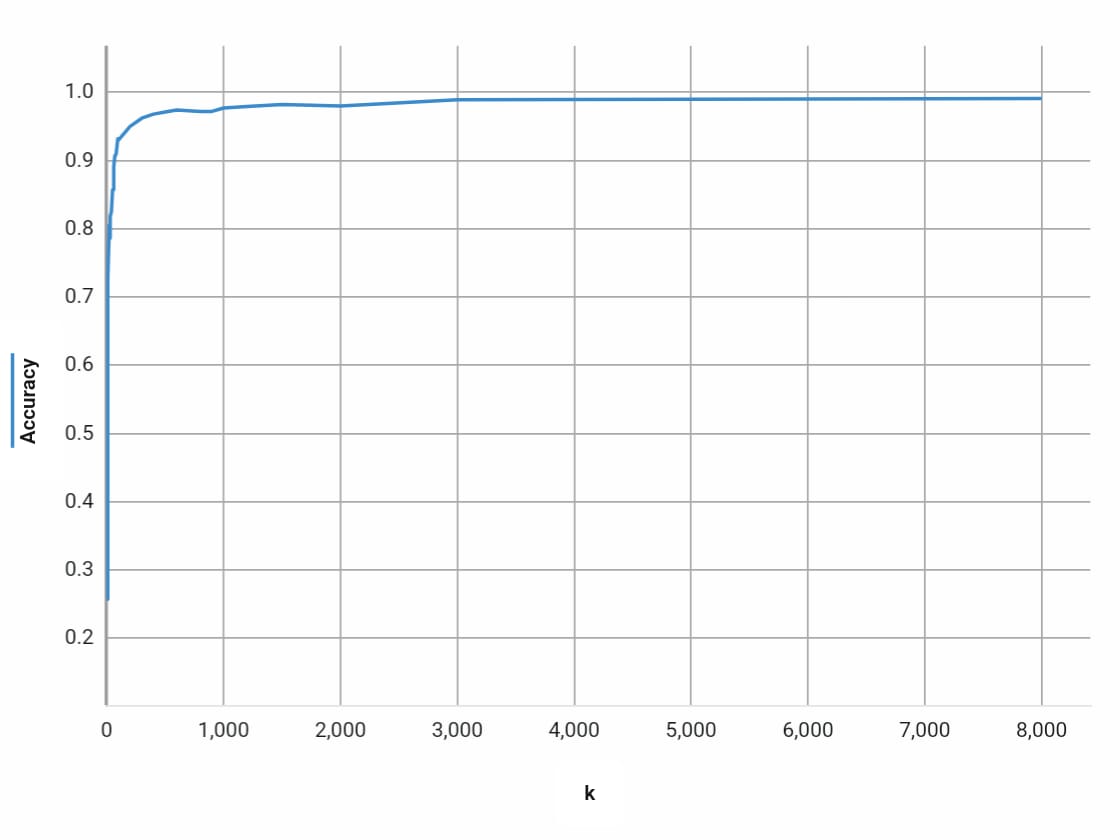


FIGURE 6

samples accuracy-k graph for dataset 3 using CNN architecture

|  |  |  |
| --- | --- | --- |
|  | k | Accuracy |
|  | 50  100  200  300  400  500  600  700  800  900  1000  1100 | 0.261  0.297  0.387  0.437  0.442  0.495  0.535  0.590  0.684  0.726  0.774  0.828 |

TABLE V

k and Accuracy values for dataset 4 using CNN architecture: The CNN model exhibits progressive performance gains, achieving an accuracy of 82.8% with 1100 samples, highlighting its ability to learn and improve accuracy with incremental sample sizes.

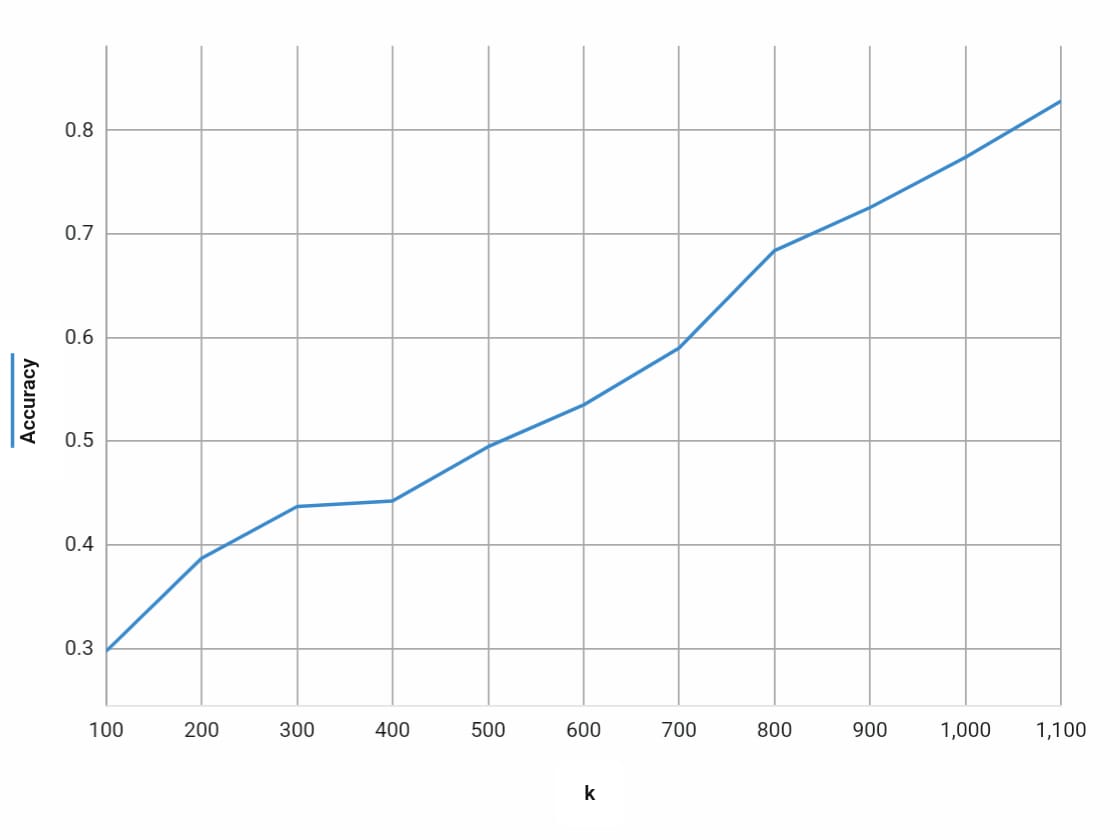


FIGURE 7

accuracy-k graph for dataset 4 using CNN architecture (the graph depicts the model’s accuracy growth, showing an upward trend as the number of samples increases)

**VI. DISCUSSION**

The dataset-dependent nature of the relationship between data quantity and model performance stands as a core finding of this study. While augmenting data samples often yields accuracy improvements, the existence of critical points and saturation phenomena emphasize the necessity of a context-sensitive approach. These insights, combined with the efficacy of low-shot learning techniques, indicating that even with restricted data volumes, notable strides can be achieved in the realm of sign language detection. The analysis of the accuracy improvement pattern for the ResNet-50 architecture on Dataset 2 (Figure 5) further highlights the model's responsiveness to data increments, with initial gains in accuracy reaching a point of stabilization. A similar behavior is observed when comparing these results with the CNN model for Dataset 2 (Figure 4). Hence, they demonstrate consistency in their responses to the increasing amount of data. This comparison underscores that, while differing slightly in accuracy values, the broader trends remain steadfast. Further, the demanding nature of the ResNet-50 architecture places significant resource constraints, resulting in execution delays and infeasibility for these datasets. These limitations highlight the intricate interplay between model complexity, dataset size, and the computational resources required.

**VII. CONCLUSION**

The central question that we aimed to address in this study is, "How does the application of low-shot learning and deep learning models impact the accuracy and efficiency of sign language detection, and what is the minimum data requirement for effective performance?" For the first part of this question, we found that that bringing together low-shot learning and deep learning models boosts accuracy and efficiency in sign language detection. For the second part, our study found that the minimum data varied across different datasets.

To summarize, employing a CNN model, we explored the impact of varying data quantities on accuracy, revealing that substantial data volumes are not an absolute requirement. Transitioning to ResNet-50, we encountered computational constraints, highlighting the intricate interplay between model complexity and resources. Our findings underscore the significance of strategic data selection, efficient architectures, and computational awareness. Ultimately, this study redefines data-efficient model development, paving a path toward enhanced sign language translation and detection through a harmonious fusion of advanced technology and better data practices.

For future studies, refinement of the training process through techniques like data augmentation, which diversifies training examples, and integration of a thorough validation set for overfitting monitoring, would contribute to a more robust and precise model training and assessment process. For more comprehensive understanding of the interplay between model architecture, dataset complexity, and data quantity, exploring techniques like model ensembles, transfer learning, or fine-tuning could be considered in the future.

**References**

[1] Ananthanarayana, Tejaswini & Srivastava, Priyanshu & Chintha, Akash & Santha, Akhil & Landy, Brian & Panaro, Joseph & Webster, Andre & Kotecha, Nikunj & Sah, Shagan & Sarchet, Thomastine & Ptucha, Raymond & Nwogu, Ifeoma. (2021). Deep Learning Methods for Sign Language Translation. ACM Transactions on Accessible Computing. 14. 1-30. 10.1145/3477498.

[2] S. He, "Research of a Sign Language Translation System Based on Deep Learning," 2019 International Conference on Artificial Intelligence and Advanced Manufacturing (AIAM), Dublin, Ireland, 2019, pp. 392-396, doi: 10.1109/AIAM48774.2019.00083.

[3] Maxim Dupont, “Manual Data Annotation vs. Augmented Annotation: Pros and Cons”, https://www.labelvisor.com/manual-data-annotation-vs-augmented-annotation-pros-and-cons/

[4] International Journal of Research in Engineering, Science and Management Volume 5, Issue 6, June2022https://www.ijresm.com | ISSN (Online): 2581-5792.

[5] Kouichi Murakami and Hitomi Taguchi, Gesture recognition using recurrent neural networks. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 237–242, 1991.

[6] Jong-Sung Kim, Won Jang, and Zeungnam Bien. A dynamic gesture recognition system for the korean sign language (ksl). IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26(2):354–359, 1996.

[7] Chan-Su Lee, Zeungnam Bien, Gyu-Tae Park, Won Jang, Jong-Sung Kim, and Sung-Kwon Kim. Real-time recognition system of korean sign language based on elementary components. In Proceedings of 6th International Fuzzy Systems Conference, volume 3, pages 1463–1468. IEEE, 1997.

[8] Manjula B Waldron and Soowon Kim. Isolated asl sign recognition system for deaf persons. IEEE Transactions on rehabilitation engineering, 3(3):261–271, 1995.

[9] Ming-Hsuan Yang, Narendra Ahuja, and Mark Tabb. Extraction of 2d motion trajectories and its application to hand gesture recognition. IEEE Transactions on pattern analysis and machine intelligence, 24(8):1061–1074, 2002.

[10] Wang, M., et al., Scene classification of high-resolution remotely sensed image based on ResNet. Journal of Geovisualization and Spatial Analysis, 2019. 3(2): p. 1-9.

[11] Saini B, Venkatesh D, Chaudhari N, et al. (2023) A comparative analysis of Indian sign language recognition using deep learning models. Forum for Linguistic Studies 5(1): 197–222. DOI: 10.18063/fls.v5i1.1617.

[12] Wang, F., Li, C., Zeng, Z. et al. Cornerstone network with feature extractor: a metric-based few-shot model for chinese natural sign language. Appl Intell 51, 7139–7150 (2021).

[13] P. T. Krishnan and P. Balasubramanian, "Detection of Alphabets for Machine Translation of Sign Language Using Deep Neural Net", 2019 International Conference on Data Science and Communication (IconDSC), pp. 1-3, 2019.

[14] Ismail Hakki Yemenoglu, A.F.M.Shahen Shan, Haci Ilhan "Deep Convolutional Neural Networks-Based Sign Language Recognition System," IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON).

[15] LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. Proc. IEEE 1998, 86, 2278–2324.

[16] Chen, T.; Xu, R.; He, Y.; Wang, X. A gloss composition and context clustering based distributed word sense representation model. Entropy 2015, 17, 6007–6024.

[17] Krizhevsky, A.; Sutskever, I.; Hinton, G.E. Imagenet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012; pp. 1097–1105.

[18] Nair, V.; Hinton, G.E. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning, Haifa, Israel, 21–24 June 2010; pp. 807–814.

[19] Aldhaheri, A.; Lee, J. Event detection on large social media using temporal analysis. In Proceedings of the Computing and Communication Workshop and Conference, Las Vegas, NV, USA, 9–11 January 2017; pp. 1–6.

[20] Hijazi, S.; Kumar, R.; Rowen, C. Using Convolutional Neural Networks for Image Recognition. 2015.

[21] Albelwi, S.; Mahmood, A. A Framework for Designing the Architectures of Deep Convolutional Neural Networks. Entropy 2017, 19, 242.

[22] Hochreiter, Sepp. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems. 6. 107-116. 10.1142/S0218488598000094.

[23] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[24] Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-architecture-of-ResNet-50-model\_fig4\_349717475 [accessed 20 Aug, 2023]

[25] <https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758>

[26] <https://www.kaggle.com/datasets/datamunge/sign-language-mnist>

[27]<https://www.kaggle.com/datasets/prathumarikeri/indian-sign-language-isl>

[28]<https://www.kaggle.com/datasets/muhammadkhalid/sign-language-for-numbers>

[29] Xin, M., Wang, Y. Research on image classification model based on deep convolution neural network. J Image Video Proc. 2019, 40 (2019).

[30] Nalinda Kulathunga, Nishath Rajiv Ranasinghe, Daniel Vrinceanu, Zackary Kinsman, Lei Huang, Yunjiao Wang. Effects of the Nonlinearity in Activation Functions on the Performance of Deep Learning Models.

[31] Kouretas, I.; Paliouras, V. Hardware Implementation of a Softmax-Like Function for Deep Learning. Technologies 2020, 8, 46.

[32] A. Victor Ikechukwu, S. Murali, R. Deepu, R.C. Shivamurthy, ResNet-50 vs VGG-19 vs training from scratch: A comparative analysis of the segmentation and classification of Pneumonia from chest X-ray images, Global Transitions Proceedings, Volume 2, Issue 2, 2021.

[33] Harriet L. Dawson, Olivier Dubrule, Cédric M. John, Impact of dataset size and convolutional neural network architecture on transfer learning for carbonate rock classification, Computers & Geosciences,Volume 171, 2023, 105284, ISSN 0098-3004.