Final Project Report - Applied Artificial Intelligence

Anonymous submission

Paper ID

1. Abstract

Sign language is an important part of everyday life for people with special abilities. Our sign language recognition project is based on the recognition of alphabets and numerals in a picture. In this project, we analyse and convert ASL (American Sign Language) pictures into understandable sentences by recognising alphabets (alphanumeric) using multiple CNN architectures. We have investigated the variance associated with varying numbers of pictures on which a model is trained using different datasets with different number classification classes available. We observed that our best model came out to be ResNet (99.18% accuracy) with our biggest dataset in similar setup environment. With large number of images we achieved better results proving the worth of well processed data and why we need large volumes to achieve better results in CNN models. These findings will be useful in bridging the communication gap and employing deep learning technology to develop simpler image-to-text applications.

2. Introduction

2.1. Problem Statement and Challanges

More than 430 million individuals worldwide suffer from disabling hearing impairment, and over 1.5 billion suffer from hearing loss[7]. People with special needs have always struggled to communicate and express themselves. The general public's failure to acquire sign language is causing a communication vacuum, and as a result, the specially-abled have limited access to school and nearly no opportunities for gainful employment. The primary reason for the communication gap is that individuals are either unwilling or unable to grasp sign language[2].

The associated challenges with respect to the problem comes with the obligation to generate high accuracy of our models so our detection can generate coherent sentences and meaningful words and communicate the meaning well.This issue statement has been plagued by noisy and indistinct photos with various backgrounds. Furthermore, configurations such as various light conditions, hand size, and the degree of unpredictability in motions make it more difficult to solve.We obtained roughly 99.19% accuracy on our largest model utilising Resnet-18 with the correct configuration and model, which is quite important towards our targeted aim. We've incorporated numerous preprocessing techniques, which we'll go through later, and we've standardised our images to provide our model a decent foundation to train on. We aim to maximise our results with limited resources so that our models may be used extensively in situations where processing power and other circumstances may not be favourable, as in our case. To date, technology has driven most of the study and development of automated sign language recognition; the availability of hardware such as datawear, as well as the development of classification algorithms, has brought us to a point where automatic recognition seems realistic. Although significant progress has been achieved in basic sign capture and categorization, it is time to reconsider the nature of sign language and what is necessary to really identify sign language. Two major new components have been identified: facial expression recognition and sign language grammar, however others, such as body position, will ultimately have to be included.

But we should not be overly optimistic. Real-time 100% recognition of full-vocabulary sign language is at least as challenging as a similar level of recognition of natural speech. Yet even to be able to recognize signs in a more restricted way based on a limited vocabulary, for instance - could be most valuable, and more attainable in the foreseeable future[18].

Sensor-based and image-based detection of hand gestures are the two most used methods. The purpose of this project is to analyse and convert ASL (American Sign Language) pictures into understandable sentences by recognising alphabets and numerals using an image-based technique based on multiple CNN architectures. We encountered obstacles when working on this subject, such as some signs having a similar look and form, therefore we attempted to normalise the data using mean and variance, and datasets were standardised with 224x224 pixel size for generality of picture and batch size. Our chosen models have a good validation accuracy since our datasets 1, 2 and 3 contain fewer noisy pictures from which we can simply extract ad-

ditional features. We used a confusion matrix to examine the output/predictions, calculating the F1 score, precision, recall, and plotting the accuracy and loss curves, and then gathered insights about how our model is performing. Furthermore, we explored the variance associated with varying the number of images on which a model is trained using different datasets with multiple classes (alphanumeric). These findings will be valuable in bridging the communication gap[10] and developing simpler image-to-text communication using deep learning technologies.

The focus of our current problem statement is detecting alphanumeric gestures and not the whole sentences which could be an extension of our current progress made during this project.

2.2. Related Work

Recognizing signs in real time requires the use of shape representation algorithms that can quickly compute and accurately characterise the shape of the hand. To make the system reliable, these strategies should also offer translational, rotational, and scaling invariance. Thalange Et.al[28], in her paper proposed various feature extraction on a plain background for recognising signs for numbers from 0 to 9 and obtained 74.69%, 82.92%, 87.94, and 98.17% of recognition rates using Statistical Measures Technique, Orientation Histogram Technique, COHST (Combined Orientation Histogram and Statistical Technique), and Wavelet Features Technique respectively.

Ss Et.al[24], in his research has put in an effort to produce images and videos in different environments ranging from black backgrounds to colourful backgrounds. Here ASL alphabet gestures yields an accuracy of 97.5% for all 24 ASL alphabets comparing with 24 obstructed ASL alphabet gestures of 70.83

Tangsuksan Et.al[27] explains B spline curvature concept and geometric invariance method for recognizing a hand gesture of 24 ASL alphabets is used on images captured by the web camera. The results came out were just satisfactory.

ASL recognition basically depends on what is the translation of any hand gesture and posture included in sign language, and continues/deals from sign gesture until the step of text generation to the ordinary people to understand deaf people. To detect any sign, a feature extraction step is a crucial phase in the recognition system. It plays the most important role in sign recognition. They must be unique, normalised, and preprocessed. Many algorithms have been suggested to solve sign recognition ranging from traditional machine learning (ML) algorithms to deep learning algorithms. On the other hand, few researchers have focused on SLID. SLID is the task of assigning a language when given a collection of hand gestures, postures, movements, and facial expressions or movements. The term "SLID" arose in the last decade as a result of many attempts to globalise and identify a global sign language. The identification process is considered as a multiclass classification problem. There are many contributions in this field with prior surveys. To the best of our knowledge, no prior works have surveyed SLID in previous Literature. This shortage was due to the need for experts who can explain and illustrate many different SLs to researchers. Also, this shortage is due to the distinction between any SL and its spoken language (i.e., ASL is not a manual form of English and does not have a unified written form)[25].

3. Methodology

3.1. Datasets

The initial structures of our datasets were as follows:

Datasets	**Dataset1[3]**	**Dataset2[22]**	**Dataset3[23]**
Images	87,000	2,515	25,300
Classes	29	36	36
Images/class	2970	70	700; '0'(841)/'t'(650)
Size(Pixels)	200x200	200x200	400x400
Format	.jpg	.jpeg	.jpeg

To speed up the calculation, we downsampled the image data in Datasets 1 and 2. We were previously doing it on the original size, which needed exorbitant resources and effort.

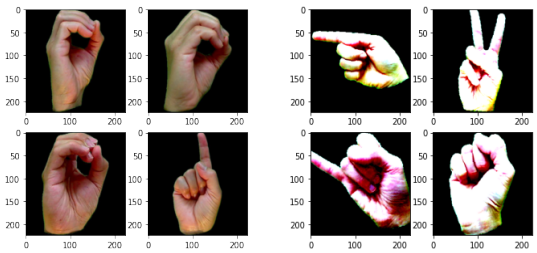
Post preprocessing the dataset structures are as follows:

Datasets	**Dataset1[3]**	**Dataset2[22]**	**Dataset3[23]**
Images	13050	2,515	7200
Classes	29	36	36
Images/class	450	70	200
Size(Pixels)	224x224	224x224	224x224
Format	.jpg	.jpeg	.jpeg

Our datasets are all from the open source site Kaggle. We resized all of our images to 224x224 pixels in all datasets, indicating that we upscaled the image size in datasets 1 and 3 while decreased the size in dataset 2. This was done in line with the prexisting CNN architectures.

We also used data augmentation techniques such as random horizontal flip (p=0.25) and rotation (degrees=15) to increase the unpredictability of the data for better training outcomes. We used this arrangement without any more substantial augmentation because our datasets were pristine with low noise, great variance, and background isolation. Feature Extraction is one of the most important task in our project as it is directly proportional to the performance of our model.

Our train, validation and test breakdown was based on a 70-30 rule throughout the process, which is 30 for testing, 60 for training, and 10 for validation.



3.2. CNN Models

3.2.1 ShuffleNet v2

ShuffleNet v2 [17] is a convolutional neural network that is tuned for speed rather than indirect measures such as FLOPs.It uses pointwise group convolutions, bottleneck-like structures, and a channel shuffle mechanism. The overall ShuffleNet architecture is presented in Table 1 [26]. ShuffleNet is composed of a stack of ShuffleNet units grouped into three stages. In Table 1, the group number g controls the connection sparsity of pointwise convolutions. At the same time, g is assigned to be different numbers, so the output channels can be computed and evaluated to ensure the total computational costs are approximately the same(140 MFLOPs). The network can be flexibly customised to achieve the required level of complexity. Simply apply a scale factor s to the number of channels to achieve this. For example, if the networks in Table 1 are labelled as "ShuffleNet 1," then "ShuffleNet s" indicates multiplying the number of filters in ShuffleNet 1 by s, resulting in an overall complexity of around s squared times ShuffleNet 1.

3.2.2 ResNet 18

ResNet-18 [8] is an 18-layer deep convolutional neural network. A pre-trained version of the network trained on over a million photos from the ImageNet database can be loaded, We have utilised the exact option for transfer learning purpose. The network accepts 224-by-224 image input[19, 12].

3.2.3 Mobilenet v2

The MobileNetV2 [20] architecture is based on an inverted residual structure, with the input and output of the residual block being thin bottleneck layers. Unlike traditional residual models, which use expanded representations in the input, MobileNetV2 filters features in the intermediate expansion layer using lightweight depthwise convolutions.

t: expansion factor, c: number of output channels, n: repeating number, s: stride. 3×3 kernels are used for spatial convolution.In typical use, the primary network (width multiplier 1, 224×224)[29].

3.2.4 Why did we choose shufflenet?

ShuffleNet v2 is clearly quicker than other networks, particularly on GPU. At 500MFLOPs, for example, ShuffleNet v2 is 58% faster than MobileNet v2 and 63% faster than ShuffleNet v1. Hence fits within our computing resource constraints[13].

3.2.5 Why did we choose MobileNet?

Although this comes on the second position to shufflenet, still it is one of the best we could find based on the research we did.For example, with width multiplier of 1.4, MobileNetV2 (1.4) outperforms ShuffleNet (×2), and NASNet with faster inference time.[30]. It complied with our resource limits as well. Also, it will help us do our ablation study and how they compare to the other ones we choose.

3.2.6 Why did we choose ResNet?

ResNets are one of the most efficient Neural Network Architectures as they help in maintaining a low error rate much deeper in the network. It is commonly regarded as one of the finest models for image classification. Furthermore, our experiment has demonstrated that we have achieved the greatest results with Resnet for our largest dataset. Also, because ResNet is a DNN with an identity shortcut link, models can pass/skip one or more layers. This allows us to train on hundreds of layers without compromising performance.

On top of the reasons mentioned above, a few more reasons that affected our decision making were affordability and time constraint.

Initially, we chose VGG as our model, but during training, we discovered that exorbitant processing power and time resources were being spent. So, after consulting with team members, TA and the Professor, we decided to go with Shufflenet.

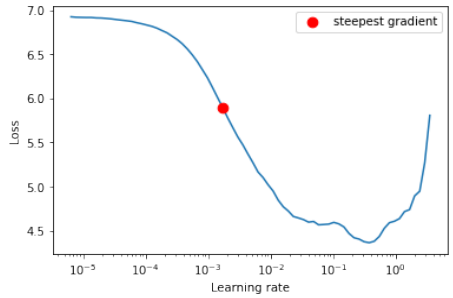
3.2.7 Complexity and Flop Count[22,23]

Model	Parameter	GMACs	GFLOPS
Resnet	11.69M	1.82GMac	3.64GFLOPs
ShuffleNet	2.28M	150.6MMac	301.2MFLOPs
MobileNet	3.5M	320.24MMac	640.48MFLOPs

For the Flops study, we used a library called ptflops to calculate the number of parameters, model complexity, and flops, as indicated in the table above. We also made a file named FLOPS_calculation.ipnyb in which we investigated layer by layer parameter count, FLOPs (floating point operations), and MACs (Multiply-Accumulate).

3.3. Optimization Algorithm

We are utilising the LRfinder library tool to determine the optimal learning rate[6] for our model. LRFinder[15] uses a method where the learning rate goes from a very small value to a very large value (i.e. from 1e-6 to 100), causing the training loss to start with a plateau, descend to some minimum value, and eventually explode. This typical behaviour can be displayed on a plot (like in the image below) and used to select an appropriate range for the learning rate, specifically in the region where the loss is decreasing. To determine the ideal learning rate, we are performing hyperparameter tuning on multiple learning rates. Due to time restrictions, we choose to implement it for the smallest dataset rather than the best dataset accuracy. To see the effect of altering learning rates, we kept the epochs, batchsize, and optimizer constant.



We employed the Adam optimizer for our optimization strategy. According to research, Adam exceeds all other optimizers in terms of experimental performance, including AdaGrad, SGD, RMSP, and others. This type of optimizer is useful for large datasets. As we know, this optimizer is a combination of the Momentum and RMSP optimization methods[1]. The Adam optimizer outperforms all other optimization techniques in terms of performance, computation time, and tweaking parameters. Because of this, Adam is recommended as the default optimizer for the vast majority of applications.

For all weight updates, stochastic gradient descent maintains a single learning rate (called alpha), which does not fluctuate during training. Each network weight (parameter) has its own learning rate that is adjusted as learning unfolds. Individual adaptive learning rates for distinct parameters are calculated using estimations of the gradient's first and second moments. Adam is characterised as integrating the benefits of two prior stochastic gradient descent improvements. AdaGrad, a per-parameter learning rate that increases performance on problems with sparse gradients (e.g., natural language and computer vision issues)[4]. Root Mean Square Propagation (RMSProp), which additionally maintains per-parameter learning rates based on the average of recent magnitudes of gradients for the weights (e.g. how quickly it is changing). This implies that the method performs well on both online and non-stationary issues (e.g. noisy). Adam recognises the advantages of AdaGrad and RMSProp. Rather than modifying the parameter learning rates based on the average first moment (the mean), as in RMSProp, Adam additionally uses the average of the gradients' second moments (the uncentered variance). The technique, in particular, computes an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 regulate the decay rates of both moving averages.

We used a variety of strategies to observe, validate, and tweak our models, the results of which may be seen later in this report. However, to provide a quick overview of the assessment measures we employed, charts such as Training Accuracy Plot, Training Loss Plot, Validation Accuracy Plot, and Validation Loss Plot. They offer a broad indication of how effectively (or poorly) our model is doing. The loss value indicates how poorly or well a model operates after each optimization iteration[21]. Accuracy is a measure of how close your model's prediction is to the actual data.

We used a confusion matrix as an assessment measure on all 9 model combinations (3 models X 3 datasets), 2 transfer learning models, and all other investigations. We use it to monitor the performance of our models.

Other evaluation metrics include precision, recall, F1-measure to evaluate performance. Precision is one indicator of a model's performance, the quality of a positive prediction made by the model[14]. The recall is calculated as the ratio between the numbers of Positive samples correctly classified as Positive to the total number of Positive samples[5]. The F-score or F-measure is a measure of a test's accuracy. It is calculated from the precision and recall of the test[9].

TSNE is used here to visualise the structure of our image data of high dimension to separate the classes from one another.

3.3.1 Adam Configuration Paramters

alpha- Also referred to as the learning rate or step size. The proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training.**beta1-** The exponential decay rate for the first moment estimates (e.g. 0.9).**beta2-** The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. NLP and computer vision problems).**epsilon-** Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8).

Further, learning rate decay can also be used with Adam. The paper uses a decay rate alpha = alpha/sqrt(t) updated each epoch (t) for the logistic regression demonstration[**cite19**].We can see that the major deep

learning packages often utilise the suggested default parameters[11].

Torch: learning_rate=0.001, beta1=0.9, beta2=0.999, epsilon=1e-8

4. Results

4.1. Expermient Setup

We downloaded our datasets from Kaggle and linked them to our colab notebooks using the official API. We downsampled our datasets after collecting them to fit our computing capabilities. We then used augmentation techniques to standardise our data images before feeding them to our chosen models. The performance was then evaluated using a variety of measures, including the confusion matrix, F1-score, precision, recall, accuracy and loss plots, TSNE visualisation, and complexity computations. We kept our code in a github repository and ran our models on Google Colab using its GPU-based resources. We also kept a separate Google Drive for sharing colab files and weights between the team members. For all 11 models, we used the Adam optimizer with a learning rate of 0.001 and a batch size of 32. The learning rate was chosen since it is the default in the Pytorch package. We picked a batch size of 32 since a batch size of 64 ran into CPU restrictions and did not result in any increase in training speed. With 16 batch sizes, model training was taking longer than we could afford. For Adam, the optimizer's results are typically better than those of any other optimization technique, it has a faster calculation time, and requires less parameters for customization, making it an ideal optimizer for us. We used the cross entropy function for the loss function, which is also known as logarithmic loss, log loss, or logistic loss. Cross entropy loss is a statistic used in machine learning to assess how well a classification model performs. The loss (or error) is expressed as a number between 0 and 1, with 0 being the ideal model. Because of restricted GPU resources and time restrictions, we ran our model across 30 epochs[16].

4.2. Main Results

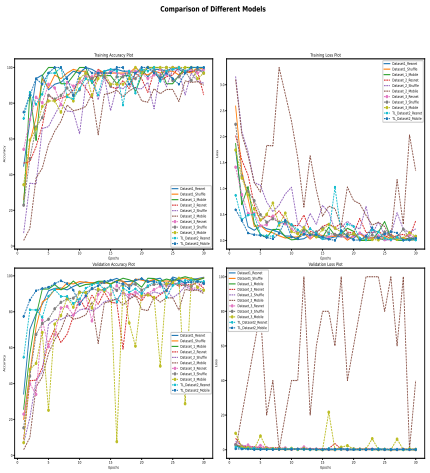
This table is based on common grounds with the same parameters as mentioned above in our experimental setup.

Dataset	Models	Time/Epoch	Accuracy	F1 Score	Precision	Recall
Dataset1	ResNet	20.74	99.18	0.99	0.99	0.99
	MobileNet	21.74	98.64	0.98	0.98	0.98
	ShuffleNet	21.48	98.41	0.98	0.98	0.98
Dataset 2	ResNet	1.31	90.05	0.88	0.92	0.89
	MobileNet	1.37	94.03	0.94	0.95	0.94
	ShuffleNet	1.22	94.42	0.94	0.95	0.93
	ResNet (TL)	1.3	97.08	0.97	0.97	0.97
	MobileNet (TL)	1.28	96.02	0.96	0.96	0.96
Dataset 3	ResNet	10.57	93.37	0.93	0.93	0.93
	MobileNet	10.38	90.78	0.9	0.92	0.9
	ShuffleNet	9.2	96.48	0.96	0.96	0.96

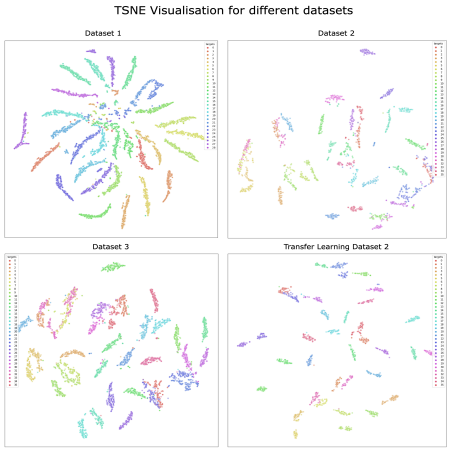
We discovered that our best model was ResNet (99.18% accuracy) with our largest dataset, dataset 1, and that all other models continually performed better across this dataset. We produced better outcomes with a larger number of images and training data, demonstrating the value of data and why huge volumes are required to achieve better results in CNN models.

Transfer learning produced the greatest results (97.1%) in dataset 2. It aided us in fine-tuning our model more efficiently and quickly on the smallest dataset and supplementing our lack of training images in the scenario, since other non-transfer learning models have lower accuracy and suffer due to a lack of data and less number of epochs in this circumstance.

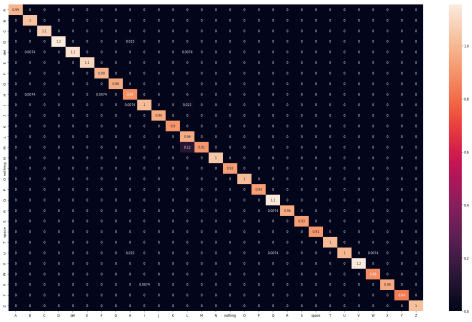
The best model in dataset 3 was shufflenet (96.5% accuracy). This is due to the behaviour of the architecture provided by shufflenet, which reduces parameters and computation while retaining accuracy.The performance of the ShuffleNet is attributable to its efficient architecture rather than depth; this can be demonstrated by examining shallower ShuffleNet versions (0.5X), which nevertheless outperform their MobileNet equivalents.The findings for other models with the same setup were unsatisfactory, indicating that there may be a lack of diversity in the images and that most images are just variations made with some logic.



The figure above is provided for all MobileNet models. The t-SNE visualisation clearly shows that our accuracies and image feature partitioning are working as planned, and most of the classes are just marginally apparent to differentiation. In the visualisation, images are clustered tightly with similar seeming hand motions, or at least near them. The findings were unexpected because transfer learning performed the best because it was defined on pretrained weights of ImageNet, which is also a multi classification issue and hence gives good results. Other models are doing admirably, as predicted. All models are operating admirably, with accuracies ranging from 90 to 99 percent.



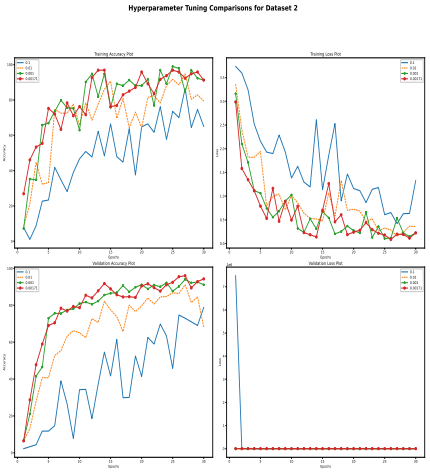
The diagram above is based on the MobileNet models on all the datasets and one of it's transfer learning variation . The t-SNE visualisation clearly reveals that our accuracies and image feature partitioning are operating as expected, and that most of the classes are apparent to differentiation. Images in the visualisation are closely plotted with identical appearing hand gestures, or at least near them. For our lat t-SNE plot which is based on transfer learning t he results were unexpectely good as other model on the similar dataset has lower performance in comaprison. Transfer learning worked well because it was defined on ImageNet pretrained weights, which is likewise a multi classification issue and hence produces good results.



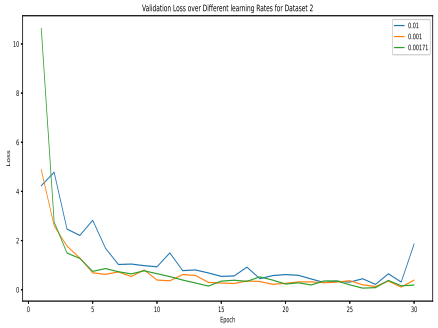
This confusion matrix displays the best model we trained (in our example, Resnet for Dataset 1), and owing to space constraints, we are only displaying one of our best models. The confusion matrix is included in all of our model training files. The Confusion Matrix allowed us analyse our performance by class, and we saw how two similar-looking hand movements confused our models. We also demonstrated how well our datasets perform in terms of F1-Score, Precision, and Recall, taking into account both predicted and ground truth values.

4.3. Ablative Study

Using our dataset 2, we tweaked our learning rate hyperparameters on shufflenet for our ablation investigation. We selected 0.1, 0.01, 0.001, and the optimal learning rate of 0.00171 that we discovered using LRFinder. We maintained all other parameters constant, which are consistent with our other training model combination. The outcomes are as follows:



As you can see, the fourth image is flat. It is because the learning rate of 0.1 finds local minima and has a very large loss during the first few epochs, therefore overshadowing the findings resulting into overfitting.



The graphic above shows that, with the exception of the 0.1 learning rate observation, the other three learning rates behave as predicted.As seen in the graphs above, the best performing learning rate is 0.00171 (discovered with LRFinder).

We actually performed learning rate hyper parameter tuning for all three models with dataset 2, and we also wanted to perform batch-size, epochs, and varying image size ablation studies, but due to pagesize constraints in the report and time constraints, we are only observing the shufflenet ablation study on learning rate.

References

[1] *Adam*. URL: https://optimization.cbe.cornell.edu.

[2] Nisha Advani et al. "A Survey on Communication Gap between Hearing and Speech Impaired Persons and Normal Persons". In: (2013). 10-04-2022.

[3] Akash. *Asl alphabet*. Apr. 2018. URL: https://www.kaggle.com/datasets/grassknoted/asl-alphabet.

[4] Jason Brownlee. *Gentle introduction to the adam optimization algorithm for deep learning*. Jan. 2021. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[5] Jason Brownlee. *How to calculate precision, recall, and F-measure for imbalanced classification*. Aug. 2020. URL: https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/.

[6] Davidtvs. *Davidtvs/pytorch-LR-finder: A learning rate range test implementation in pytorch*. URL: https://github.com/davidtvs/pytorch-lr-finder.

[7] *Deafness and hearing loss*. URL: https://www.who.int/health-topics/hearing-loss.

[8] *Deep Network designer*. URL: https://www.mathworks.com/help/deeplearning/ref/resnet18.html.

[9] *F-score*. Dec. 2022. URL: https://en.wikipedia.org/wiki/F-score.

[10] Ahmed KASAPBAŞI et al. "DeepASLR: A CNN based human computer interface for American Sign Language recognition for hearing-impaired individuals". In: *Computer Methods and Programs in Biomedicine Update* 2 (2022), p. 100048. ISSN: 2666-9900. DOI: https://doi.org/10.1016/j.cmpbup.2021.100048. URL: https://www.sciencedirect.com/science/article/pii/S2666990021000471.

[11] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv.org* (Jan. 2017). URL: https://arxiv.org/abs/1412.6980.

[12] *Lecture Slides*. URL: https://moodle.concordia.ca/moodle/pluginfile.php/5709730/mod_resource/content/1/08_cnn_architectures.pdf.

[13] Ningning Ma et al. "ShuffleNet V2: Practical guidelines for efficient CNN Architecture Design". In: *SpringerLink* (Jan. 1970). URL: https://link.springer.com/chapter/10.1007/978-3-030-01264-9_8.

[14] *Machine learning*. URL: https://c3.ai/glossary/machine-learning/.

[15] Simona Maggio. *The Learning Rate Finder technique: How reliable is it?* URL: https://blog.dataiku.com/the-learning-rate-finder-technique-how-reliable-is-it.

[16] Saurav Maheshkar. *What is cross entropy loss? A tutorial with code*. Sept. 2021. URL: https://wandb.ai/sauravmaheshkar/cross-entropy/reports/What-Is-Cross-Entropy-Loss-A-Tutorial-With-Code--VmlldzoxMDA5NTMx.

[17] *Papers with code -shufflenet V2 explained*. URL: https://paperswithcode.com/method/shufflenet-v2.

[18] *Progress in sing language recognition*. URL: https://www-users.cs.york.ac.uk/~alistair/publications/html/GW97.html.

[19] *Resnet-18 Architecture*. URL: https://www.researchgate.net/figure/ResNet-18-Architecture_tbl1_322476121.

[20] Mark Sandler et al. "MobileNetV2: Inverted residuals and linear bottlenecks". In: *arXiv.org* (Mar. 2019). URL: https://arxiv.org/abs/1801.04381.

[21] Sonam, Shrutiparna, and Vinita. *How to interpret "loss" and "accuracy" for a machine learning model*. Sept. 2019. URL: https://intellipaat.com/community/368/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model.

[22] Sovrasov. *Sovrasov/flops-counter.pytorch: Flops counter for convolutional networks in Pytorch Framework*. URL: https://github.com/sovrasov/flops-counter.pytorch.

[23] Sovrasov. *What is the relationship between gmacs and GFLOPS? issue 16 Sovrasov/Flops-counter.pytorch*. URL: https://github.com/sovrasov/flops-counter.pytorch/issues/16.

[24] Shivashankara Ss and Dr.Srinath S. "American Sign Language Recognition System: An Optimal Approach". In: *International Journal of Image, Graphics and Signal Processing* 10 (Aug. 2018). DOI: 10.5815/ijigsp.2018.08.03.

[25] Ahmed Sultan et al. *Sign language identification and recognition: A comparative study*. Jan. 2022. URL: https://www.degruyter.com/document/doi/10.1515/comp-2022-0240/html.

[26] Synced. *ShuffleNet: An extremely efficient convolutional neural network for mobile devices*. Aug. 2017. URL: https://medium.com/syncedreview/shufflenet-an-extremely-efficient-convolutional-neural-network-for-mobile-devices-72c6f5b01651.

[27] Watcharin Tangsuksant, Suchin Adhan, and Chuchart Pintavirooj. "American Sign Language recognition by using 3D geometric invariant feature and ANN classification". In: (Nov. 2014), pp. 1–5. DOI: 10.1109/BMEiCON.2014.7017372.

[28] Asha Thalange and S.K. Dixit. "COHST and Wavelet Features Based Static ASL Numbers Recognition". In: *Procedia Computer Science* 92 (Dec. 2016), pp. 455–460. DOI: 10.1016/j.procs.2016.07.367.

[29] Sik-Ho Tsang. *Review: MOBILENETV2-light weight model (image classification)*. Aug. 2019. URL: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c.

[30] Sik-Ho Tsang. "Review: MOBILENETV2-light weight model (image classification)". In: *Medium* (Aug. 2019). URL: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c.