International Journal of Academic Engineering Research (IJAER)
ISSN: 2643-9085
Vol. 6 Issue 8, August - 2022, Pages: 9-19

Classification of Sign-Language Using Deep Learning - A Comparison between Inception and Xception models

Tanseem N. Abu-Jamie and Pror.Dr. Samy S. Abu-Naser

Department of Information Technology,
Faculty of Engineering and Information Technology,
Al-Azhar University - Gaza, Palestine

Abstract*: there is a communication gap between hearing-impaired people and those with normal hearing, sign language is the main means of communication in the hearing-impaired population. Continuous sign language recognition, which can close the communication gap, is a difficult task since the ordered annotations are weakly supervised and there is no frame-level label. To solve this issue, we compare the accuracy of each model using two deep learning models, Inception and Xception . To that end, the purpose of this paper is By attempting to translate sign language using deep learning models, we focused in this paper a comparing between Inception and Xception algorithm and compared it to previous papers in which we used VGG19[1], mobileNet [2], where we get in the Inception and Xception algorithm on the degree of Accuracy 99.32%, 99,43% respectively. number of images (29000 in the dataset in size 75*75 pixel) and the data split training data into training dataset (70%) and validation dataset(15%) and testing dataset(15%) and 20 epoch.*

Keywords: *sign language Classification, sign language, Deep Learning, Classification, Detection*

1. INTRODUCTION:

People who have trouble speaking or hearing use sign language as a way of communication. Non-verbal movements in sign language are used by people to convey their thoughts and feelings. However, non-signers find it very challenging to grasp, hence it is necessary to have trained sign language interpreters present during legal and medical consultations as well as educational and training events. The need for translating services has grown during the previous five years. There are now additional methods available, such as video remote human interpretation using high-speed Internet connections. Thus, they will offer a service that is simple to use but has significant drawbacks for sign language interpretation.

We employ an ensemble of two models to recognize sign language gestures in order to address this. We use the custom American Sign Language Dataset. The dataset has 29 categories contend(letter, space, delete, nothing) .For simplicity, we get a dataset from kaggel website. We propose to use a CNN (Convolutional Neural Network) model named Inception and xception to classification categories for Sign Language Recognition.

The rest of this research is organized as follows: Section 2 gives an overview of the literature study performed, Section 3 discusses the methodology used in two models. shows the experimental and result in the models used are described in Section 4. Finally, Section 5 contains the conclusion.

1. REVIEW OF LITERATURE SURVEY

Over the previous 2 centuries, a number of researchers have worked in the field of sign language recognition.

In this paper [3] they evaluated results on a multi-user data set with two scenarios: one with all known users and one with an unseen user. they achieved 99% recall and precision on the first, and 77% recall, and 79% precision on the second.her method is also capable of real-time sign classification and is adaptive to any environment or lightning intensity.

In this paper [4]their proposed network was able to achieve an accuracy of 90.3 % which is comparable to other works including those that used depth images in addition to RGB images. their network was also able to achieve prediction rates of 50 to 100 Hz which makes it capable of real-time prediction..

In this paper [5] they present an optimal approach to recognize BdSL —Bangla Sign language (BdSL) in real-time. First, theywe have developed BdSL Infinite dataset, which consists of 2,000 images of 37 different signs. Using this dataset, a convolutional neural network (CNN) based model is trained using Xception architecture that achieves 98.93% accuracy over the test-set, with response time of 48.53 ms on average. To the best of our knowledge, their proposed method outperforms all existing BdSL recognition methods in terms of both accuracy and speed.

In [6] In this study, the data set was created by taking 10223 images for 29 letters in the Turkish Sign Language Alphabet. Images are made suitable for education by using image enhancement techniques. In the final stage of the study, classification processes on images were carried out by using CapsNet, AlexNet and ResNet-50, DenseNet, VGG16, Xception, InceptionV3, NasNet, EfficentNet, Hitnet, Squeezenet architectures and TSLNet, which was designed for the study. When the deep learning models were examined, it was found that CapsNet and TSLNet models were the most

successful models with 99.7% and 99.6% accuracy rates, respectively.

In [7] in this study an Arabic sign language recognition using two concatenated deep convolution neural network models DenseNet121 & VGG16 is presented. The pre-trained models are fed with images, and then the system can automatically recognize the Arabic sign language. To evaluate the performance of concatenated two models in the Arabic sign language recognition, the red-green-blue (RGB) images for various static signs are collected in a dataset. The dataset comprises 220,000 images for 44 categories: 32 letters, 11 numbers (0:10), and 1 for none. For each of the static signs, there are 5000 images collected from different volunteers. and their get the best convolutional neural networks (CNN) model in feature extraction and classification Arabic sign language is the DenseNet121 for a single model using and DenseNet121 & VGG16 for multi-model using.

In[8] this paper, presents results obtained by retraining and testing this sign language gestures dataset on a convolutional neural network model using Inception v3. The model consists of multiple convolution filter inputs that are processed on the same input. The validation accuracy obtained was above 90% This paper also reviews the various attempts that have been made at sign language detection using machine learning and depth data of images. It takes stock of the various challenges posed in tackling such a problem, and outlines future scope as well..

in this paper [9] they have proposed a deep dense inception residual network for three-class brain tumor classification. they have customized the output layer of Inception ResNet v2 with a deep dense network and a softmax layer. The deep dense network has improved the classification accuracy of the proposed model. The proposed model has been evaluated using key performance metrics on a publicly available brain tumor image dataset having 3064 images. her proposed model outperforms the existing model with a mean accuracy of 99.69%. Further, similar performance has been obtained on noisy data.

The paper [10] proposes they goal is to obtain an effective way for early diagnosis of skin cancer by classifying our dataset images as benign or malignant. they dataset consists of 2437 training images, 660 test images and lastly 200 validation images. ResNet-101 and Inception-v3 deep learning architectures are used for the classification task. Once the acquired results are examined, an accuracy rate of 84.09% is get in ResNet-101 architecture, and an accuracy rate of 87.42% is get in Inception-v3 architecture.

In this paper [11] they present an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depthwise separable convolution operation (a depthwise convolution followed by a pointwise convolution). thier to propose a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depthwise separable convolutions. We show that this architecture, dubbed Xception, slightly outperforms Inception V3 on the ImageNet dataset (which Inception V3 was designed for), and significantly outperforms Inception V3 on a larger image classification dataset comprising 350 million images and 17,000 classes. Since the Xception architecture has the same number of parameters as Inception V3, the performance gains are not due to increased capacity but rather to a more efficient use of model parameters.

2. METHODOLOGY

The proposed system for Sign Language Recognition System comprises of data to Kaggle the dataset contains sets of images in jpeg extension and prepare data to apply the architecture CNN by Inception and Xception models we used just 1000 image for every class so we used 29000 as shown in the table1 image data split training data into training dataset (70%) and validation dataset (15%) and testing dataset(15%) Training and testing the dataset.

Due to the size similarities, we compare Xception to the Inception V3 architecture: There are nearly as many parameters in Inception V3 and Xception.

Table 1:number of datasets for every class

class	Number of image
(A-Z)letter	1000 image for every letter
space	1000
delete	1000
nothing	1000
total	29000

A. Dataset

The proposed system employs deep learning techniques for data recognition in the training phase, By using a two models in deep learning Inception and Xception which is critical to the overall system. In the first step, the number of images was very large, so we decided to only use 29000 of the 87,000 images. This data contains 29 classes (A to Z), spaces, delete, and nothing .

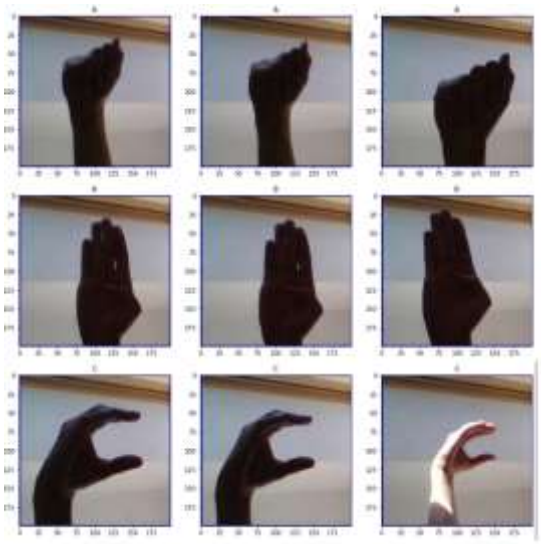

Figure 1:some of latters used in dataset

B. Deep Learning

A machine learning technology called deep learning enables computers to learn from examples in the same manner that people do. Recently, deep learning has attracted a lot of research attention, and for good reason. It involves achieving goals that were previously unreachable. In deep learning, a computer model learns to carry out categorization tasks directly from images, text, or sound.

Deep learning models can achieve cutting-edge accuracy, sometimes even outperforming human ability. Models are trained using multilayer neural network topologies and a vast amount of labeled data.

The proposed system's goal is to recognize the sign that is represented by the sign and 29 signs considered for the experiment so it is having multiple category, classes are provided as 29 target class for the sign language recognition system.

C. Inception

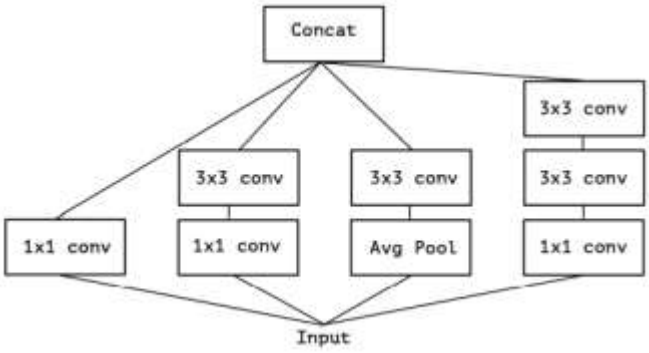

Figure 2: A canonical Inception module (Inception V3)

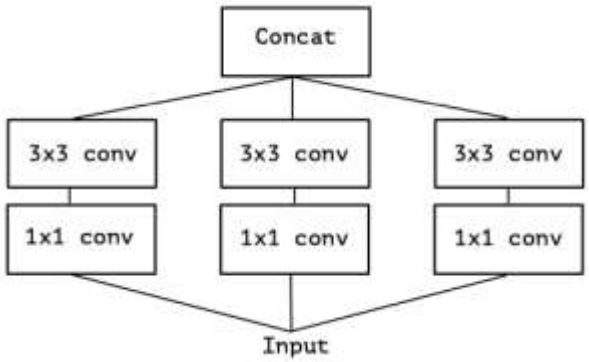

Figure 3:A canonical Inception module (Inception V3)

A single convolution kernel is faced with concurrently mapping cross-channel correlations and spatial correlations since a convolution layer attempts to learn filters in a 3D space with two spatial dimensions (width and height) and a channel dimension. The purpose of the Inception module is to explicitly factor this process into a set of operations that would separately examine cross-channel correlations and spatial correlations in order to make it simpler and more effective. The typical Inception module, to be more precise, first maps the input data into 3 or 4 separate spaces that are smaller than the original input space, looking at cross-channel correlations via a set of 1x1 convolutions, and then maps all correlations

in these smaller 3D spaces via regular 3x3 or 5x5 convolutions.

Figure 2 is an illustration of this. Cross-channel correlations and spatial correlations are essentially sufficiently separated, according to the central tenet of Inception, that it is preferable not to map them together.

Consider a condensed form of an Inception module that simply employs a single convolution size (like 3x3) and excludes an ordinary pooling tower (figure 3). It is possible to rewrite this Inception module as a significant 1x1 convolution followed by spatial convolutions that operate on non – overlapped output channel segments (figure 4).

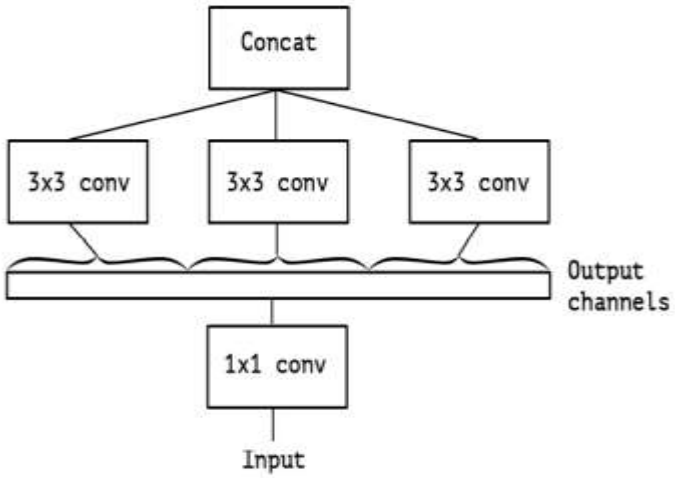

Figure 4:A strictly equivalent reformulation of the simplified Inception module

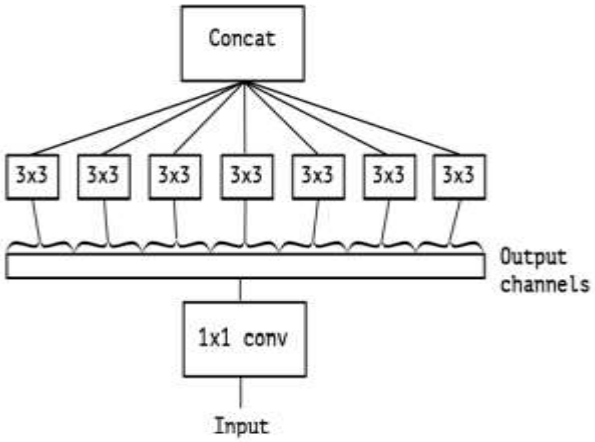

Figure 5: An "extreme" version of our Inception module, with one spatial convolution per output channel of the 1x1 convolution.

Based on this stronger premise, a "extreme" version of an Inception module would first map cross-channel correlations using a 1x1 convolution, and then it would map the spatial correlations of each output channel independently. In figure 5, this is displayed. We note that the depthwise separable convolution, an operation that has been employed in neural networks, and this extreme form of an Inception module are nearly identical.

D. Xception

We suggest an architecture for a convolutional neural network made completely of depth-separable convolution layers. In summary, we claim that cross-channel and spatial correlations can be completely dissociated from one another when mapped into feature maps of convolutional neural networks. We call our suggested architecture Xception, which stands for "Extreme Inception," because this theory is a more robust version of the one underpinning the Inception architecture.

A complete description of the specifications of the network is given in figure 5. The Xception architecture has 36 convolutional layers forming the feature extraction base of the network.

The Xception architecture can be summed up as a linear stack of residually connected depthwise separable convolution layers. This makes defining and changing the architecture relatively simple.

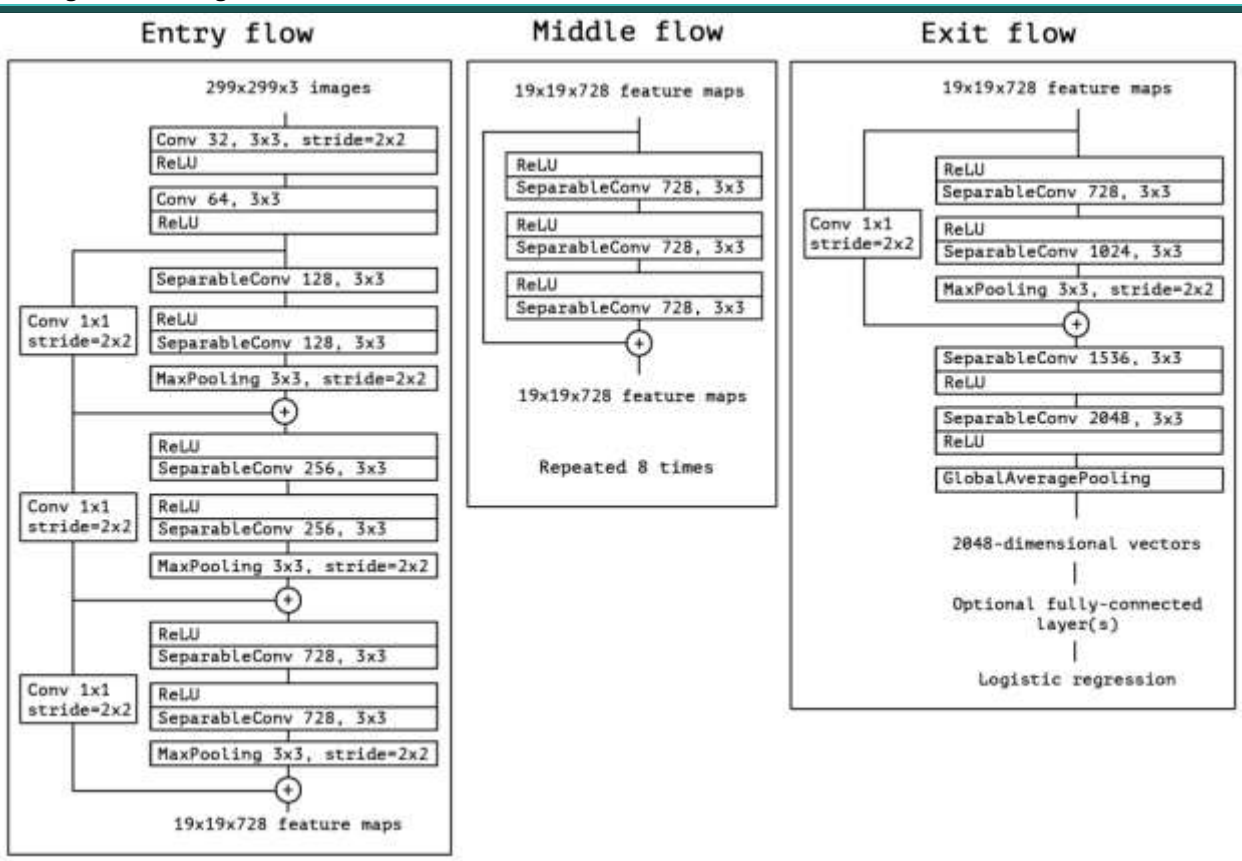

Figure 6: The Xception architecture: the data first goes through the entry flow, then through the middle flow which is repeated eight times, and finally through the exit flow. Note that all Convolution and SeparableConvolution layers are followed by batch normalization (not included in the diagram). All SeparableConvolution layers use a depth multiplier of 1 (no depth expansion).

E. Training Algorithm

According to Table 1, the pretrained CNN InceptionV3 and Xception models is used. Preprocessing images with a resolution of 75x75x3 pixels. The following parameters were used to fit the model: the learning rate is 0.0001, the activation is softmax, the loss is categorical crossentropy, Adam is the optimizer, and the epoch is 20.

Model	Number of image	Loading Time (seconds)	Testing Accuracy (Percent)
Inception	29000	394.3861s	100.00%%
Xception	29000	234.6671s	99.51%

Table 2: MobileNet Model Evaluation based on number of image, Loading Time, and testing Accuracy

3. EXPERIMENTAL RESULTS

1. load all images in memory

we use filter to choose the best 1000 images for each class and resize to (75*75px). So the result in this step :

1. Training data shape: (29000, 75, 75, 3)
2. Training labels shape: (29000, 29)

2. **Split dataset:** training data into training dataset (70%) and validation dataset(15%) and testing dataset(15%)by import train_test_split from sklearn.model_selection

3. call the MobileNet pre-trained Model

we make function contain a InceptionV3 and Xception model use(weights= 'imagenet', include_top=False, input_shape= (75,75,3)) then make MaxPooling2D and Dense then used compile (Adam(lr=0.00001), loss='categorical_crossentropy', metrics=['accuracy']) So the result in this table 3.

Table 3:parameters for every model

model / params	Total params	Trainable params	Non-trainable params
Xception	20,920,901	20,866,373	54,528
InceptionV3	21,862,205	21,827,773	34,432

4. Train model

All networks were implemented using the TensorFlow framework and platform googlecolab The results of a convolution neural network's by Xception and InceptionV3 loss rate in the training and test sets after 20 repetitions are shown in Figure 7 and Figure 8 respectively. Which suggests that the convolutional neural network successfully learned the input and can act as a good model for understanding sign language but VGG16 100% in the previous paper [1] was better from mobileNet 95,41%[2] accuracy

```
Epoch 1/20
40/40 [==============================] - 37s 444ms/step - loss: 3.2949 - accuracy: 0.0705 - val_loss: 3.6466 - val_accuracy: 0.0421
Epoch 2/20
40/40 [==============================] - 18s 440ms/step - loss: 2.8786 - accuracy: 0.2057 - val_loss: 3.1197 - val_accuracy: 0.1356
Epoch 3/20
40/40 [==============================] - 16s 393ms/step - loss: 2.3944 - accuracy: 0.3729 - val_loss: 2.7074 - val_accuracy: 0.3149
Epoch 4/20
40/40 [==============================] - 16s 397ms/step - loss: 1.9378 - accuracy: 0.5121 - val_loss: 2.2650 - val_accuracy: 0.5087
Epoch 5/20
40/40 [==============================] - 16s 404ms/step - loss: 1.4943 - accuracy: 0.6459 - val_loss: 1.7900 - val_accuracy: 0.6694
Epoch 6/20
40/40 [==============================] - 16s 401ms/step - loss: 1.1101 - accuracy: 0.7627 - val_loss: 1.3173 - val_accuracy: 0.7954
Epoch 7/20
40/40 [==============================] - 16s 402ms/step - loss: 0.7354 - accuracy: 0.8467 - val_loss: 0.8896 - val_accuracy: 0.8662
Epoch 8/20
40/40 [==============================] - 16s 401ms/step - loss: 0.5073 - accuracy: 0.9081 - val_loss: 0.5866 - val_accuracy: 0.9193
Epoch 9/20
40/40 [==============================] - 16s 401ms/step - loss: 0.3712 - accuracy: 0.9312 - val_loss: 0.3950 - val_accuracy: 0.9506
Epoch 10/20
40/40 [==============================] - 16s 398ms/step - loss: 0.2673 - accuracy: 0.9555 - val_loss: 0.2760 - val_accuracy: 0.9660
Epoch 11/20
40/40 [==============================] - 16s 403ms/step - loss: 0.1988 - accuracy: 0.9672 - val_loss: 0.2161 - val_accuracy: 0.9720
Epoch 12/20
40/40 [==============================] - 16s 397ms/step - loss: 0.1597 - accuracy: 0.9744 - val_loss: 0.1798 - val_accuracy: 0.9768
Epoch 13/20
40/40 [==============================] - 16s 398ms/step - loss: 0.1294 - accuracy: 0.9828 - val_loss: 0.1512 - val_accuracy: 0.9789
Epoch 14/20
40/40 [==============================] - 16s 403ms/step - loss: 0.1093 - accuracy: 0.9835 - val_loss: 0.1292 - val_accuracy: 0.9818
Epoch 15/20
40/40 [==============================] - 16s 403ms/step - loss: 0.0952 - accuracy: 0.9854 - val_loss: 0.1156 - val_accuracy: 0.9832
Epoch 16/20
40/40 [==============================] - 16s 402ms/step - loss: 0.0807 - accuracy: 0.9893 - val_loss: 0.1042 - val_accuracy: 0.9844
Epoch 17/20
40/40 [==============================] - 16s 400ms/step - loss: 0.0696 - accuracy: 0.9895 - val_loss: 0.0924 - val_accuracy: 0.9867
Epoch 18/20
40/40 [==============================] - 19s 478ms/step - loss: 0.0681 - accuracy: 0.9916 - val_loss: 0.0894 - val_accuracy: 0.9871
Epoch 19/20
40/40 [==============================] - 16s 397ms/step - loss: 0.0590 - accuracy: 0.9922 - val_loss: 0.0805 - val_accuracy: 0.9883
Epoch 20/20
40/40 [==============================] - 15s 364ms/step - loss: 0.0471 - accuracy: 0.9943 - val_loss: 0.0854 - val_accuracy: 0.9869
Time elapsed in seconds: 385.9759624004364
```

Figure 7: Loss and Accuracy rate to Xception model

```
Epoch 1/20
40/40 [==============================] - 38s 407ms/step - loss: 3.3409 - accuracy: 0.0613 - val_loss: 3.5065 - val_accuracy: 0.0634
Epoch 2/20
40/40 [==============================] - 9s 215ms/step - loss: 3.1304 - accuracy: 0.1553 - val_loss: 3.3014 - val_accuracy: 0.0906
Epoch 3/20
40/40 [==============================] - 9s 219ms/step - loss: 2.9030 - accuracy: 0.2746 - val_loss: 3.1461 - val_accuracy: 0.1834
Epoch 4/20
40/40 [==============================] - 12s 294ms/step - loss: 2.5813 - accuracy: 0.4287 - val_loss: 2.8752 - val_accuracy: 0.3320
Epoch 5/20
40/40 [==============================] - 9s 223ms/step - loss: 2.1933 - accuracy: 0.5783 - val_loss: 2.4800 - val_accuracy: 0.5366
Epoch 6/20
40/40 [==============================] - 9s 233ms/step - loss: 1.8497 - accuracy: 0.6742 - val_loss: 2.0257 - val_accuracy: 0.7103
Epoch 7/20
40/40 [==============================] - 9s 232ms/step - loss: 1.4830 - accuracy: 0.7705 - val_loss: 1.5855 - val_accuracy: 0.8060
Epoch 8/20
40/40 [==============================] - 9s 232ms/step - loss: 1.1823 - accuracy: 0.8336 - val_loss: 1.2101 - val_accuracy: 0.8676
Epoch 9/20
40/40 [==============================] - 9s 236ms/step - loss: 0.9135 - accuracy: 0.8896 - val_loss: 0.9269 - val_accuracy: 0.9069
Epoch 10/20
40/40 [==============================] - 10s 250ms/step - loss: 0.7125 - accuracy: 0.9166 - val_loss: 0.6959 - val_accuracy: 0.9336
Epoch 11/20
40/40 [==============================] - 16s 406ms/step - loss: 0.5812 - accuracy: 0.9341 - val_loss: 0.5260 - val_accuracy: 0.9543
Epoch 12/20
40/40 [==============================] - 9s 221ms/step - loss: 0.4408 - accuracy: 0.9575 - val_loss: 0.4049 - val_accuracy: 0.9653
Epoch 13/20
40/40 [==============================] - 9s 222ms/step - loss: 0.3507 - accuracy: 0.9617 - val_loss: 0.3081 - val_accuracy: 0.9729
Epoch 14/20
40/40 [==============================] - 9s 224ms/step - loss: 0.2803 - accuracy: 0.9699 - val_loss: 0.2363 - val_accuracy: 0.9791
Epoch 15/20
40/40 [==============================] - 9s 232ms/step - loss: 0.2211 - accuracy: 0.9778 - val_loss: 0.1850 - val_accuracy: 0.9871
Epoch 16/20
40/40 [==============================] - 9s 222ms/step - loss: 0.1874 - accuracy: 0.9852 - val_loss: 0.1473 - val_accuracy: 0.9885
Epoch 17/20
40/40 [==============================] - 9s 230ms/step - loss: 0.1561 - accuracy: 0.9867 - val_loss: 0.1134 - val_accuracy: 0.9931
Epoch 18/20
40/40 [==============================] - 9s 225ms/step - loss: 0.1247 - accuracy: 0.9893 - val_loss: 0.0908 - val_accuracy: 0.9947
Epoch 19/20
40/40 [==============================] - 9s 234ms/step - loss: 0.1115 - accuracy: 0.9900 - val_loss: 0.0753 - val_accuracy: 0.9959
Epoch 20/20
40/40 [==============================] - 9s 228ms/step - loss: 0.0909 - accuracy: 0.9932 - val_loss: 0.0609 - val_accuracy: 0.9975
Time elapsed in seconds: 231.19872975349426
```

Figure 8: Loss and Accuracy rate to Inception model

5. Training and validation accuracy

the data used to fit the model, the training dataset Testing dataset is used to use data for purposes other than training and validating, whereas validation dataset is used during the training phase to validate the model's generalizability or to end it early.

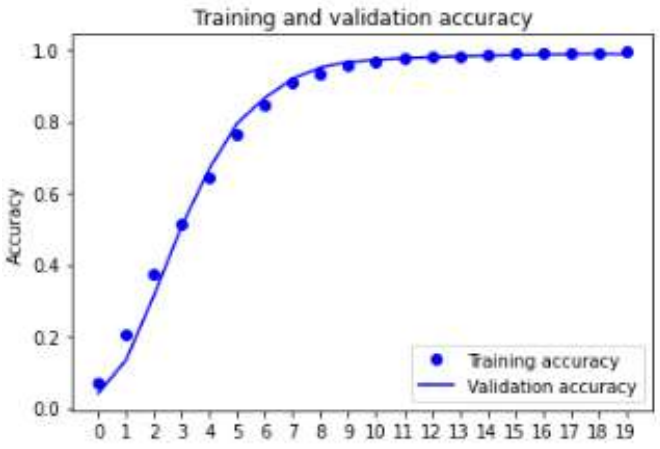

Figure 9:Training and validation accuracy to Xception model

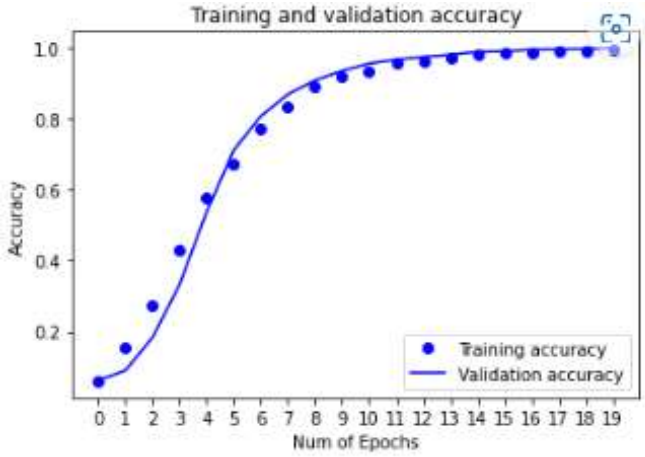

Figure 10:Training and validation accuracy to Inception model

In Figure 9 and Figure 10 show us no overfitting or under fitting.

6. Training and validation loss

The training loss is a metric that measures how well a deep learning model fits the training data.

Validation loss, on the other hand, is a metric used to evaluate the performance of a deep learning model on the validation set.

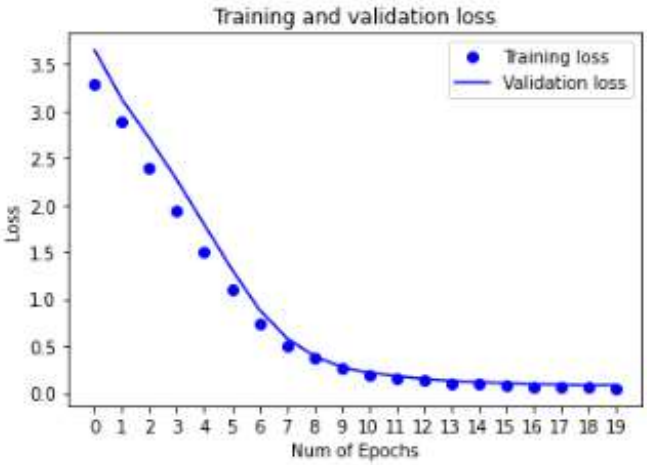

Figure 11:Training and validation loss to Xception model

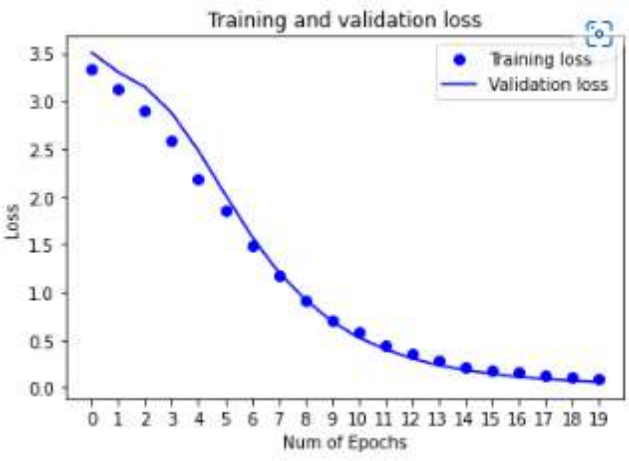

Figure 12:Training and validation loss to inception model

7. Evaluating the model on the training, validating sets

The evaluating the model process of analyzing a training model to ensure that it is delivered effectively and efficiently is known as training evaluation. Training evaluation identifies training gaps and even opportunities for model improvement By seeing each of the following values validating accuracy,validating loss ,training loss ,training accuracy If every value in training accuracy and validating accuracy has a high value, and every value in validating loss and training loss is low, it means that the model is well trained.

```
# Evaluating the model on the training, validating sets to inception model
score = model.evaluate(X_train, Y_train, verbose=0)
print("Training Accuracy: {1:.4f}, Training Loss: {0:.4f}".format(score[0], score[1]))

score = model.evaluate(X_valid, Y_valid, verbose=0)
print("Validating Accuracy: {1:.4f}, Validating Loss: {0:.4f}".format(score[0], score[1]))
```

```
Training Accuracy: 0.9979, Training Loss: 0.0446
Validating Accuracy: 0.9975, Validating Loss: 0.0609
```

Figure 13:Evaluating the model on the training, validating sets to inception model

```
# Evaluating the model on the training, validating sets to Xnception model
score = model.evaluate(X_train, Y_train, verbose=0)
print("Training Accuracy: {1:.4f}, Training Loss: {0:.4f}".format(score[0], score[1]))

score = model.evaluate(X_valid, Y_valid, verbose=0)
print("Validating Accuracy: {1:.4f}, Validating Loss: {0:.4f}".format(score[0], score[1]))
```

```
Training Accuracy: 0.9940, Training Loss: 0.0539
Validating Accuracy: 0.9883, Validating Loss: 0.0805
```

Figure 14:Evaluating the model on the training, validating sets to Xnception model

```
import time
start = time.time()

# Compute the final loss and accuracy to inception model
final_res = model.evaluate(X_test,Y_test)
print("Testing final accuracy: {1:.4f}, Testing Final loss: {0:.4f}".format(final_res[0], final_res[1]))


end = time.time()
print ("Time elapsed inb seconds:", end - start)
```

```
116/116 [==============================] - 3s 26ms/step - loss: 0.0690 - accuracy: 0.9951
Testing final accuracy: 0.9951, Testing Final loss: 0.0690
Time elapsed inb seconds: 3.4684298038482666
```

Figure 15:Compute the final loss and accuracy to inception model

```
import time
start = time.time()

# Compute the final loss and accuracy
final_res = model.evaluate(X_test,Y_test)
print("Testing final accuracy: {1:.4f}, Testing Final loss: {0:.4f}".format(final_res[0], final_res[1]))


end = time.time()
print ("Time elapsed inb seconds:", end - start)
```

```
116/116 [==============================] - 3s 26ms/step - loss: 0.0667 - accuracy: 0.9916
Testing final accuracy: 0.9916, Testing Final loss: 0.0667
Time elapsed inb seconds: 5.48798942565918
```

Figure 16:Compute the final loss and accuracy to Xception

Model / **Evaluating**	Training Accuracy	Testing Accuracy	Validating Accuracy
InceptionV3	99.79%	99.51%	99.75%
Xception	99.4%	99.16%	98.83%

Table 4: Evaluating the model on the training, validating sets

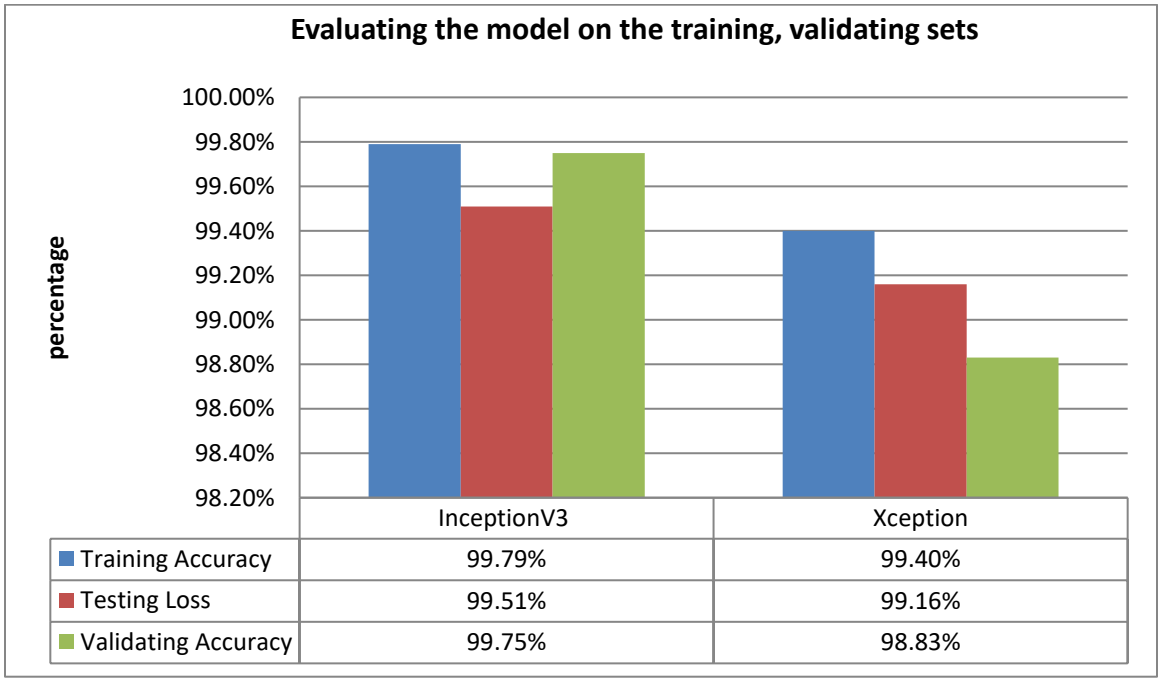

Figure 17:Evaluating the model on the training, validating sets to models

We get high testing accuracy inception model 99.51%

4. CONCLUSION

We showed how Inception modules act as a midpoint between convolutions and depthwise separable convolutions, which are at opposite ends of a discrete spectrum. We now suggest that Inception modules in neural computer vision systems be replaced with depthwise separable convolutions as a result of this observation. Based on this concept, we introduced a revolutionary architecture called Xception, which has a comparable parameter count to Inception V3. On the sign language dataset, Xception exhibits modest improvements in classification performance over Inception V3. On datasets containing 29 classes of signs, we examined the performance of the pretrained Inception and Xception models of deep learning in this study. An already-trained CNN model is used.

InceptionV3 and Xception have been improved. We used an unknown dataset to test the performance of the proposed model after it had been trained and validated. Our accuracy percentage was 99.32% to 20 epochs and 99.43% overall. This shows that our suggested model can accurately predict and classify several sign languages without error and with complete functionality, although the VGG16 achieved greater accuracy 100% in the previous paper [1] and mobileNet the Accuracy rate was achieved 95.41%[2].

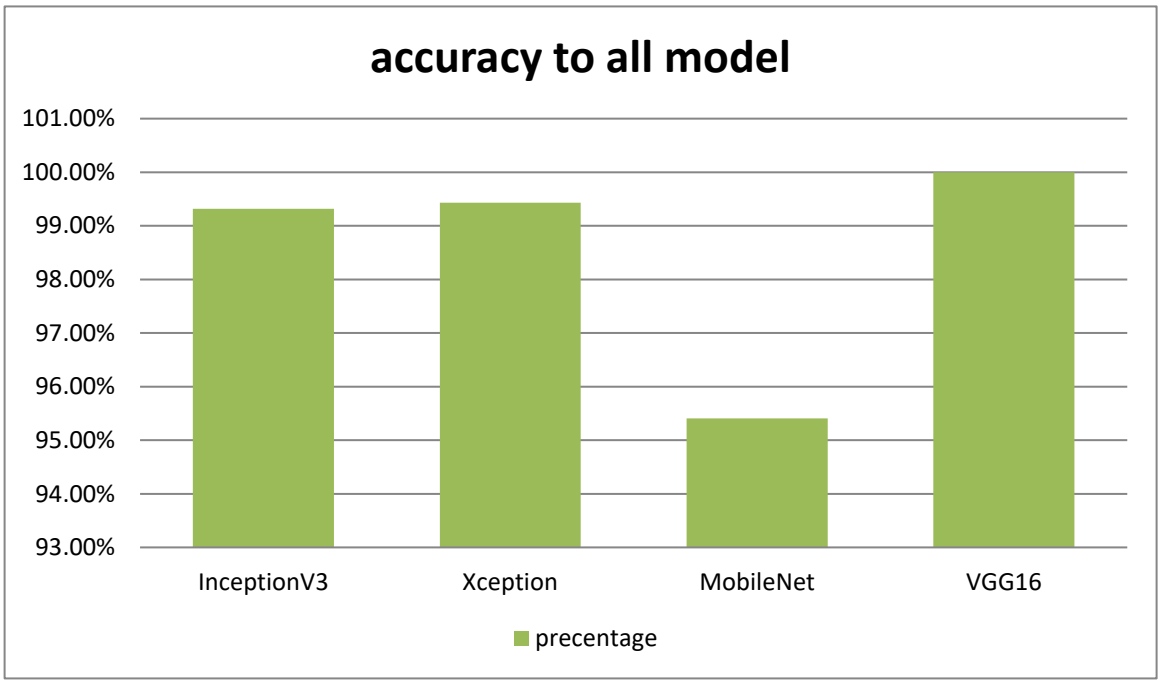

Figure 18:Comparison of all models that have been trained

References:

1. Abu-Jamie, T.N. and S.S. Abu-Naser, Classification of Sign-language Using VGG16. International Journal of Academic Engineering Research (IJAER), 2022. 6(6).
2. Abu-Jamie, Tanseem N., and Samy S. Abu-Naser. "Classification of Sign-Language Using MobileNet-Deep Learning." International Journal of Academic Information Systems Research (IJAISR) 6.7 (2022).
3. L. Rioux-Maldague and P. Giguère, "Sign Language Fingerspelling Classification from Depth and Color Images Using a Deep Belief Network," 2014 Canadian Conference on Computer and Robot Vision, 2014, pp. 92-97, doi: 10.1109/CRV.2014.20.
4. R. Daroya, D. Peralta and P. Naval, "Alphabet Sign Language Image Classification Using Deep Learning," TENCON 2018 - 2018 IEEE Region 10 Conference, 2018, pp. 0646-0650, doi: 10.1109/TENCON.2018.8650241.
5. Urmee, Progya Paromita, et al. "Real-time bangla sign language detection using xception model with augmented dataset." 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE). IEEE, 2019.
6. Aksoy, Bekir, Osamah Khaled Musleh Salman, and Özge Ekrem. "Detection of Turkish Sign Language Using Deep Learning and Image Processing Methods." Applied Artificial Intelligence 35.12 (2021): 952-981.
7. Ismail, Mohammad H., Shefa A. Dawwd, and Fakhradeen H. Ali. "Static hand gesture recognition of Arabic sign language by using deep CNN s." Indonesian Journal of Electrical Engineering and Computer Science 24.1 (2021): 178-188.
8. A. Das, S. Gawde, K. Suratwala and D. Kalbande, "Sign Language Recognition Using Deep Learning on Custom Processed Static Gesture Images," 2018 International Conference on Smart City and Emerging Technology (ICSCET), 2018, pp. 1-6, doi: 10.1109/ICSCET.2018.8537248.
9. Kokkalla, S., Kakarla, J., Venkateswarlu, I.B. et al. Three-class brain tumor classification using deep dense inception residual network. Soft Comput 25, 8721–8729 (2021).
10. A. Demir, F. Yilmaz and O. Kose, "Early detection of skin cancer using deep learning architectures: resnet-101 and inception-v3," 2019 Medical Technologies Congress (TIPTEKNO), 2019, pp. 1-4, doi: 10.1109/TIPTEKNO47231.2019.8972045.
11. Chollet, François. "Xception: Deep learning with depthwise separable convolutions." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
12. Yao, T., et al. Image recognition method of defective button battery base on improved MobileNetV1. in Chinese Conference on Image and Graphics Technologies. 2020. Springer.
13. Sinha, D. and M. El-Sharkawy. Thin mobilenet: An enhanced mobilenet architecture. in 2019 IEEE 10th annual ubiquitous computing, electronics & mobile communication conference (UEMCON). 2019. IEEE.
14. Shinde, P.P. and S. Shah. A review of machine learning and deep learning applications. in 2018 Fourth international conference on computing communication control and automation (ICCUBEA). 2018. IEEE.
15. He, K., et al. Deep residual learning for image recognition. in Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
16. Abu-Jamie, T.N., et al., Six Fruits Classification Using Deep Learning. 2022.