Multilingual Script Identification in Images

Lakshya Gupta, Prathma Rastogi
Syracuse University, Syracuse, NY, US

Abstract *- **Recognizing text with occlusion and perspective distortion in natural scenes is a challenging problem. Script identification plays an important role in analyzing images. This study will focus on identifying script in natural images and study on methods of handling images of varying aspect ratios while training models. In this study, we are presenting a dataset of around 61,768 images each containing a particular script from these six scripts: Arabic, Bangla, Latin, Korean, Japanese and Chinese for identification.We used three strategies for inputting images, which include squaring of the images to fixed size, padding the images to squares maintaining aspect ratio and with varying input sizes. The models we proposed in this study are single-layer CNN model, Inception model, ResNet model and fully-convolutional network(FCN) model and did performance evaluation of these models using metrics of Precision, Recall and f1-score. We performed a comparison of models on the basis of a weighted average of f1-score.***

Keywords --- CNN, Inception, ResNet, fully-convolutional network, Precision, Recall, f1-score

I. *INTRODUCTION*

"Over the recent years, there has been remarkable growth in multimedia data in the form of images, videos and audios. With the advancement in technologies, digital multimedia is likely to increase manifolds in the days to come." [1]

Texts within images contain rich semantic information which are beneficial for image understanding and retrieving visual information. To efficiently read texts from photography, the majority of methods follow a two-step process: text detection followed by text recognition. Text detection can greatly affect the performance and accuracy of text recognition.

In different languages, characters can have different forms and styles based on the font, size, stroke, etc. There are thousands of languages in the world, with different forms of scripts used to depict them which have their universal features still unknown [1]. In addition, a lot of images have different representation scripts of different languages based on fonts, styles, color, light variations, etc. which are still challenging to detect. There are a lot of research studies focused on detection of English language from texts but there are not so many studies which focus on detection of multilingual scripts present in images.

Figure 1

Script identification is the function to classify scripts present in images into available scripts like, Latin, Arabic, Chinese, etc. Script identification can be beneficial in solving document image analysis problems, advanced problems like font-to-font translation, handwriting trajectory recovery, staff-line removal, etc. that employ deep learning techniques. Also, images can be different in sizes which can be difficult to train using conventional neural network models. The images we have are of varying aspect ratios and if we resize these images to a fixed size, it

can be challenging to identify script due to quality reduction.

II. *RELATED WORK*

Script identification is a well described problem in document image analysis, natural scene images and videos.

[1] proposed the framework of identifying the script using Attention based CNN-LSTM networks. The images are converted into patches and fed into CNN-LSTM framework. Attention based weights are calculated using a softmax layer after LSTM. Global features are extracted from the last cell state of LSTM.

The authors of [2] approached the solution of using classifiers such as k-NN classifier and SVM classifier for multilingual script identification in natural scene images. This study mainly focused on Indic scripts with over 500 collections of images. In this study, LBP (Local Binary Pattern) [6], CS-LBP (center symmetric local binary pattern) and DLEP (directional local extrema pattern) features for each script were extracted and studied. The observed results states that SVM outperformed the k-NN classifier with 84% accuracy.

Latest approaches in this division have obtained texture level features from Local Binary Patterns [3] or Gabor filters analysis [7-9].

[11] shows the importance of deep CNN architectures with pre-trained imagenet input in classification of images.

After extensive research on the problem of script identification, the approach we proposed here is using different CNN architectures with variations of sizes in input images. One of the deep CNN architectures we used is Inception V3. In comparison to VGGNet, Inception Networks have proved to be more computationally efficient, both in terms of number of parameters generated by the network and economically cost incurred. We also used the residual network model such as ResNet50 [12] for faster training and reducing the problem of diminishing gradients.

III. *EXPLORATORY DATA ANALYSIS*

i. *Data Description*

The dataset has 61,768 images containing a single script in each image. All the images are of varying aspect ratios.

The scripts are categorized into:

- Latin: 42,629
- Korean: 4,476
- Japanese: 4,108
- Arabic: 3,505
- Bangla: 3,214
- Chinese: 2,702

The data is downloaded from source:
https://rrc.cvc.uab.es/?ch=8&com=downloads

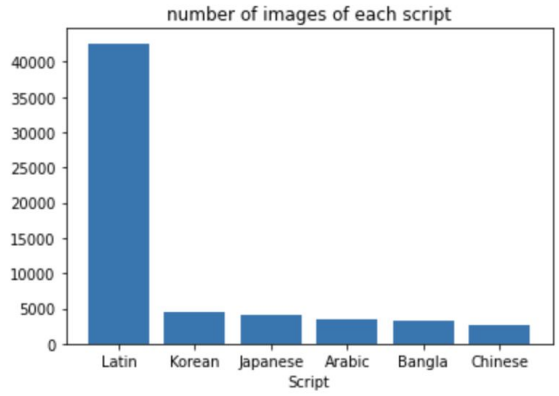

Figure 2

The distribution of classes in the data can be seen as,

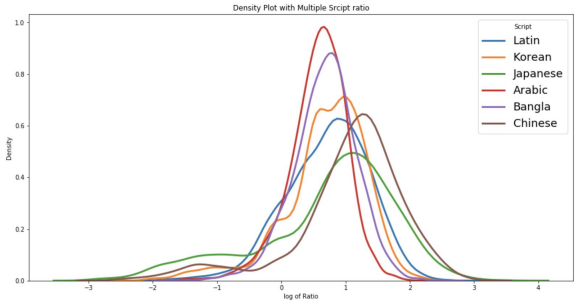

Figure 3

ii. *Data Processing*

Removed unnecessary scripts like Symbols and 3 from the data.

As shown above, data is imbalanced due to large variations in distribution of classes. We balanced the data by assigning weights to the classes using inverse frequencies.

$$W_i = T / N_i$$

where, W_i is the weight of class i, T is the total number of observations and N_i is the number of observations of class i.

As data has images of varying sizes, we created another data with padded images of square size *128x128* from the original data as in figure 6.

Some sample images from original data are:

Figure 4

Sample images from data of squared fixed size images:

Figure 5

Some sample images after padding are:

Figure 6

Further, one more dataset is created for training a fully-convolutional network model with minimum required dimension without resizing images such that input image won't get too small when applying convolutional block.

Finally, the data is split into training, test and validation samples using *ImageDataGenerator* for experimentations in models.

We used ImageNet dataset as pre-trained weights for Inception and ResNet models. The ImageNet [12] is formally a project aimed at labelling and categorizing images into 22,000 separate object categories for the purpose of computer vision research.

IV. *PROPOSED MODELS*

After performing experiments with different models, we have selected our three best performing models, i.e., single-layer CNN, Inception and ResNet. We also implemented a fully-convolutional network with variable input dimensions. For previous three models which are single-layer CNN, Inception and ResNet we have performed training using two variations of input images, i.e., one with padding and one without padding.

i. *Single-layer CNN*

To achieve the optimum CNN architecture that would fit into our model, we varied necessary parameters and tested different versions of CNN on our both versions of the dataset. The following parameters were tuned in this procedure: the base learning rate, the number of neurons in fully connected layers, the convolutional kernel sizes and dropout rate. Finally, we concluded that the CNN architecture proposed

below gives most promising results for our end-to-end model.

Type	Configuration
Input	*128 x 128 x 3* images
Conv2D	filters: *32*; filter size: *4x4*; activation: *relu* Output: *125 x 125 x 32*
MaxPooling2D	pool size: *3x3* Output: *41 x 41 x 32*
Flatten	Output: *53792*
Dense (fully connected layer)	*128* neurons; activation: *relu*
Dense (fully connected layer, output layer)	*6* neurons; activation: *softmax*

Table 1

We used the same CNN model architecture for all input variations. ReLu is applied on the convolutional layer and fully-connected layer leading to faster training and lower probability of vanishing gradient. The MaxPooling layer summarizes the most activated presence of a feature.

ii. *Inception V3*

"It is a widely used image recognition model that is 48 layers deep and has been shown to attain greater than 78.1% accuracy on the ImageNet dataset. The model itself is made up of symmetric and asymmetric building blocks, including convolutional, average pooling, max pooling, concats, dropouts, and fully-connected layers. Batchnorm is used extensively throughout the model and applied to activation inputs. Loss is computed via softmax." [14]

The high-level diagram of architecture is,

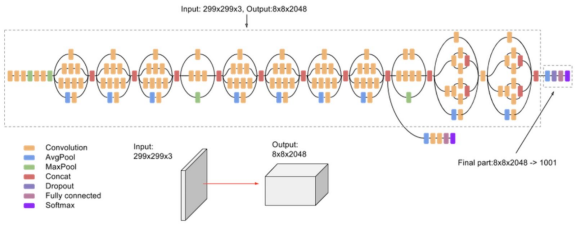

Figure 7

An auxiliary classifier is a small CNN inserted between layers during training, and the loss incurred is added to the main network loss.

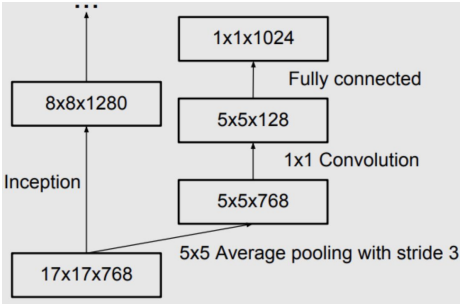

Figure 8

We trained Inception V3 with pre-trained weights from an ImageNet dataset. We used the input shape of *128x128x3*. On the Inception model output we further used regularization dropout of 0.4, and fully connected layer with 512 neurons and activation function of ReLu for faster training and lower probability of vanishing gradient. The final predictions are computed via a fully-connected layer of 6 neurons with activation function of softmax. As our problem is multi-class classification, we used the loss function of *categorical_crossentropy*.

iii. *ResNet50*

ResNet uses a technique called "residual mapping" to combat the issue of degrading with increase in depth. Building block of residual network,

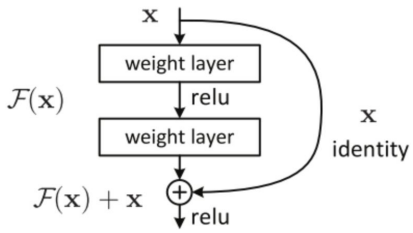

Figure 9

As compared to VGG and other deep neural networks, ResNet has far fewer filters and lower complexity during training. A shortcut connection is added that turns the network into its counterpart residual version. This shortcut connection performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no additional parameter. The projection shortcut is mathematically represented as *F(x{W}+x),* which is used to match dimensions computed by 1×1 convolutions.
Below are different ResNet architectures,

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		[3×3, 64; 3×3, 64] ×2	[3×3, 64; 3×3, 64] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3	[1×1, 64; 3×3, 64; 1×1, 256] ×3
conv3_x	28×28	[3×3, 128; 3×3, 128] ×2	[3×3, 128; 3×3, 128] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×4	[1×1, 128; 3×3, 128; 1×1, 512] ×8
conv4_x	14×14	[3×3, 256; 3×3, 256] ×2	[3×3, 256; 3×3, 256] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×6	[1×1, 256; 3×3, 256; 1×1, 1024] ×23	[1×1, 256; 3×3, 256; 1×1, 1024] ×36
conv5_x	7×7	[3×3, 512; 3×3, 512] ×2	[3×3, 512; 3×3, 512] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3	[1×1, 512; 3×3, 512; 1×1, 2048] ×3
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 10

We used ResNet50, i.e., ResNet with 50 layers for our experiments. We initialized a model with pretrained weights from an ImageNet dataset and also without pre-trained weights..The model performed better on pretrained weights, therefore, we used a model trained on pre-trained weights comparison purposes. We used *adam* optimizer with *categorical_crossentropy* loss function while compiling the model. We used a global average pooling layer on ResNet model output and the final predictions are computed via a fully-connected layer of 6 neurons with activation function of softmax.

iv. *Fully-Convolutional Network (FCN)*

The FCN does not contain any "Dense" layers, instead it contains 1x1 convolutions that perform the task of fully connected layers. The absence of dense layers makes it possible to feed variable inputs through the techniques of loading and generating batches of data in memory of variable dimensions and then training the network with these inputs. We built the FCN model by stacking convolution blocks consisting of 2D convolutional layers and required regularizations like Dropout and BatchNormalization. The regularization helps in quick convergence and prevents overfitting.

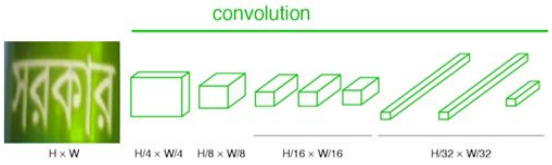

Figure 11

"After applying a convolutional block on the input, the height and width of input will decrease based on *kernel_size* and *strides.* If the input image size is too small then we might fall short of the minimum required height and width for the next convolutional block" [15]. After proper trial-error way and ensuring we get *'1, 1, num_of_filters'* as output dimensions from the last convolutional networks, we decided to use image height of 64 and width with respect to original aspect ratio of image.
Since the height and width of input images are variable, we specified input shape as *(None, None, 3)* where 3 are the number of channels in our images (RGB) which is fixed. Now, since we are not resizing our images, converting them into batches of numpy arrays becomes impossible. So we can pass each image, in the list (batch), in and that's why it is very slow. We performed this strategy but results were not that satisfying. In addition, the model is compiled using *adam* optimizer and *categorical_crossentropy* loss function. The final outputs are computed via softmax function.

V. *RESULTS*

We did performance evaluation of these models using metrics of Accuracy, Precision, Recall and f1-score. For comparison of models on the basis of a weighted average of f1-score as there is a class imbalance in the dataset.

i. ***f1-Score results on fixed size (128,128,3) images***

Scripts	Single layer CNN	Inception V3	ResNet50
Arabic	0.74	0.77	0.85
Bangla	0.62	0.79	0.80
Chinese	0.64	0.24	0.33
Japanese	0.62	0.43	0.40
Korean	0.72	0.65	0.71
Latin	0.86	0.92	0.92
Overall f1-Score	0.72	0.65	0.69
Accuracy(%)	79	83	85
No. of trainable parameters	6,887,846	21,768,352	23,546,886

Table 2

ii. ***f1-Score results on padded images***

Scripts	Single layer CNN	Inception V3	ResNet50
Arabic	0.78	0.83	0.77
Bangla	0.77	0.83	0.78
Chinese	0.70	0.40	0.13
Japanese	0.77	0.36	0.30
Korean	0.76	0.72	0.68
Latin	0.90	0.94	0.91
Overall f1-Score	0.79	0.81	0.63
Accuracy(%)	85	86	83
No. of trainable parameters	6,887,846	21,768,352	23,546,886

Table 3

iii. ***Fully-Convolutional network results***

Accuracy (%)	Precision	Recall	f1-Score
70.83	0.50	0.71	0.58

Table 4

iv. ***Models Comparison w.r.t. mean and standard deviation***

Our train data had 9095 images and we took 30 samples of 500 images each with replacements. The average and standard deviation of the weighted f1-Score on these samples are stated below in Table5. Single Layered CNN and Inception V3 have similar F1_score and standard deviation is also not high.

	Models	Avg. F1	Std. Dev. F1	Computational effort
Fixed Square Sized	*Single Layered CNN*	0.8033	0.0097	6,887,846
	Inception V3	0.8225	0.0099	21,768,352
	ResNet50	0.8569	0.0103	23,546,886
Padded Images	*Single Layered CNN*	0.8617	0.0118	6,887,846
	Inception V3	0.8564	0.0121	21,768,352
	ResNet50	0.8046	0.016	23,546,886
Varying Size	*FCN*	0.5814	0.0291	46,994

Table 5

VI. *CONCLUSION*

After our experiments, we observed that converting images into padded squared images resulted in better performance of the models. The models, i.e. ***Inception V3*** and ***Single Layered CNN*** came out to be best performance models on these inputs.
In order to consider computation efforts for the models comparison, we assumed computation effort to be proportional to number of trainable parameters and epochs. The results showed that cost of Single layered CNN is lower as compared to Inception V3, hence, it can be concluded that Single layered CNN model is an appropriate choice for classification for this type of problem.

VII. *REFERENCES*

[1] Ankan Kumar Bhunia, Aishik Konwer, Ayan Kumar Bhunia, Abir Bhowmick, Partha P. Roy, Umapada Pal, Script Identification in Natural Scene Image and Video Frame using Attention based Convolutional-LSTM Network, *Pattern Recognition* (2019), pp. 172-184.

[2] Manisha Verma, Nitakshi Sood, B. Raman, Partha P. Roy, Script Identification in Natural Scene Images: A Dataset and Texture-Feature Based Performance Evaluation, In *Proceedings of International Conference on Computer Vision and Image Processing* (2017), pp. 309-319.

[3] Nabin Sharma, Sukalpa Chanda, Umapada Pal, Michael Blumenstein, Word-Wise Script Identification from Video Frames, *IEEE 12th International Conference on Document Analysis and Recognition* (2013), pp. 369-373.

[4] B. Shi, C. Yao, C. Zhang, X. Guo, F. Huang, X. Bai, Automatic script identification in the wild, In Document Analysis and Recognition (ICDAR), *2015 13th International Conference on IEEE*, (2015) pp. 531-535.

[5] B. Shi, X. Bai, C. Yao, An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition, IEEE transactions on pattern analysis and machine intelligence. (2016)

[6] Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Pattern Analysis and Machine Intelligence, IEEE Transactions on 24(7), 971–987 (2002)

[7] T.N. Tan, Rotation invariant texture features and their use in automatic script identification, IEEE Transactions on pattern analysis and machine intelligence, 20(7) (1998) pp.751-756.

[8] W. Chan, G. Coghill, Text analysis using local energy, Pattern Recognition, 34(12) (2001) pp.2523-2532.

[9] W.M. Pan, C.Y. Suen, T.D. Bui, Script identification using steerable Gabor filters, In *Document Analysis and Recognition*, 2005. Proceedings. Eighth International Conference on IEEE, (2005) pp. 883- 887.

[10] Jason Brownlee, A Gentle Introduction to Pooling Layers for Convolutional Neural Networks, In *Deep Learning for Computer Vision* (2019).

[11] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, Rethinking the Inception Architecture for Computer Vision, In *Computer Vision and Pattern Recognition* (2015).

[12] Adrian Rosebrock, ImageNet: VGGNet, ResNet, Inception, and Xception with Keras, In *PyImageSearch* (2017).

[13] Jonathan Long, Evan Shelhamer, Trevor Darrell, Fully Convolutional Networks for Semantic Segmentation, In *Computer Vision and Pattern Recognition* (2014).

[14] Advanced Guide to Inception v3 on Cloud TPU, *https://cloud.google.com/tpu/docs/inception-v3-advanced*.

[15] Understanding and implementing a fully convolutional network (FCN), *https://towardsdatascience.com/implementing-a-fully-convolutional-network-fcn-in-tensorflow-2-3c46fb61de3b.*