# Mid-Semester Report
# ML in Fashion: Training an algorithm to recognise Indian Classical Sarees

**Sudip Kumar Kar**
School of Physical Sciences
National Institute of Science Education and Research
Jatni, Odisha , 752050
sudip.kar@niser.ac.in

**Ritadip Bharati**
School of Physical Sciences
National Institute of Science Education and Research
Jatni, Odisha ,752050
ritadip.bharati@niser.ac.in

## Abstract

Machine Learning algorithms can be trained to achieve computer vision tasks quite efficiently. The problem of identifying items of clothing has already been undertaken in some capacity before, but only for western-style garments. Here, we build a dataset of *sarees*, a specific kind of Indian apparel that is quite popular in the country, and train some well-known algorithms to recognize the kind of saree based on their images.

## 1 Introduction

The usage of neural networks to solve real-world problems has become ubiquitous in the present-day seemingly unrelated fields like fashion can also benefit from using these tools to maximize productivity. The use of machine learning in fashion is not unheard of; various open databases like *Fashion MNIST* and *DeepFashion* exist to serve this purpose, but we have yet to see any attempts at training a model to learn regional items of clothing.

An attempt has been made to train a model that can recognize various kinds of Indian *Sarees*; since no dataset containing such data exists, a custom dataset has been constructed from scratch by web scraping various Indian E-commerce websites.

## 2 Related works

The authors of [3] have compiled a dataset containing 12k images of miscellaneous Indian clothing items. Using the dataset, they have created a generative adversarial network(GAN) that tries to replicate the patterns in Indian clothes. Their work provides necessary insight into the process of web scraping and building a dataset.

VGG-net is an algorithm developed by using small filters over deep convolutional layers in [5]. A TinyVGG architecture [7] has been modeled on vgg-nets, it is well known and suitable for early implementation.
MobilenetV2 [4] is another image classification algorithm that excels in image classification using

Preprint. Under review.

Table 1: Indian Saree Dataset description

| Saree type | Total number | Training | Testing |
|---|---|---|---|
| Bandhani | 288 | 231 | 57 |
| Ilkal | 476 | 382 | 94 |
| Kasavu | 1536 | 1229 | 307 |
| Sambalpuri | 2388 | 1911 | 477 |
| Total | 4688 | 3753 | 935 |

low-end machines such as mobile devices. With some added layers, MobilenetV3 [2] has been an important improvement over MobilenetV2. ECANet [6] puts an *Efficient Channel Attention* layer in different CNN image classification algorithms such as MobilenetV2, ResNet etc.

## 3 Baseline Algorithm

The neural networks that are most suited for this problem are Convolutional Neural Networks(CNNs). These networks take advantage of convolution to flesh out hidden details in an image that are otherwise hard to perceive. The baseline algorithm is based on TinyVGG, which consists of four units of layers. The first unit is composed of two layers: the first one is a convolutional layer, and the second one is a rectified linear unit(ReLU) activation layer. The second unit is based on three layers: the first one being a convolutional layer, the second one being a ReLU layer, and the third being a max pooling layer; this layer collects the maximum value that lies within a window and constructs a new layer. These units are repeated again in the same order to give the entirety of the TinyVGG architecture.

## 4 Experimental Details

The dataset of Indian Classical Sarees is custom-made by web scraping. It contains over 4.5k images of four kinds of Indian Sarees, details are listed in Table. (1). For training purposes, this data undergoes a train-test split of 80-20, after which the data is arranged in the format prescribed by the `pytorch ImageFolder` dataloader. The code of all the experiments depicted below and the ones that were used for data scraping can be found in the Github repository.

### 4.1 TinyVGG Experiments

The architecture used in this series of experiments on Tiny VGG is identical to the one described in [7] with the exception that in the final layer, a small change is made to reflect the difference in the number of classes i.e., the final activation layer takes in 31360 features and gives out a $4\times1$ tensor this is then passed through a linear activation layer to classify the input image.

This model is then trained with Cross entropy loss as the criterion and Adam as the optimizer, in five epochs, the loss went down significantly for training data, without overfitting(refer Fig.(1). A separate set of experiments revealed a sort of upper boundary to the testing accuracy at 90%, irrespective of the number of epochs.

### 4.2 MobilenetV3 and Experiments

**Discussion**: MobilenetV3 is an advancement over MobilenetV2, a lightweight image classification algorithm operable on mobile devices. The algorithm differentiates itself from other algorithms in 3 blocks. First and foremost of them is a depth-wise separable convolution block[4], the second one is an inverted residual block, and the third one is a Squeeze and excite layer.
**depth-wise separable convolution**: Unlike traditional convolutions, which apply a single filter to all channels of an input volume, depth-wise convolution applies separate filters to each input channel. This means that each filter only processes information within a single channel, allowing the model to learn more specialized features for each channel. By reducing the number of parameters required for each filter, depth-wise convolutions are often more computationally efficient than traditional convolutions. In the present case, three channels are present, viz. Red, Green, and Blue for each image.

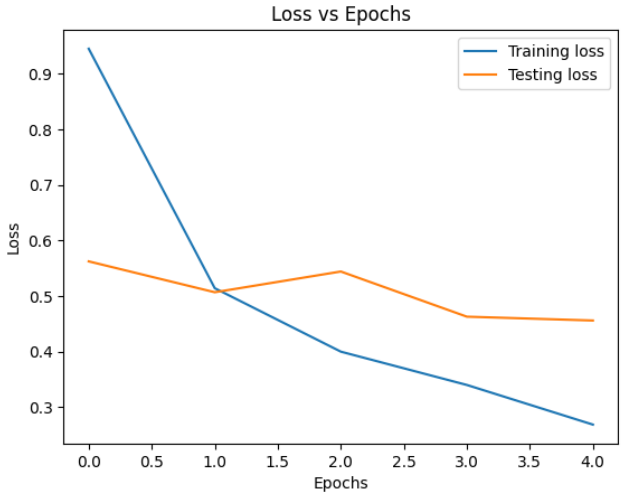


Figure 1: Loss trend over epochs for TinyVGG model

Point-wise convolutions apply a $1 \times 1$ filter to each element of the input volume, essentially performing a weighted sum of the input channels at each spatial location. This allows the model to learn linear combinations of the input features, creating new, higher-level features that can capture complex relationships between different input channels. This combination of depth-wise and point-wise convolutions is often referred to as a "depth-wise separable convolution". It is a powerful technique for reducing computational cost and improving the performance of CNNs.
A linear bottleneck is a type of network layer that consists of a point-wise convolution followed by a depth-wise convolution followed by another point-wise convolution. The first point-wise convolution reduces the number of input channels, the depth-wise convolution processes the reduced set of channels, and the second point-wise convolution expands the number of output channels.
**Inverted residual block**: The residual block and the inverted residual block are both building blocks commonly used in CNNs. The key difference between them lies in their structure and functionality. A Residual block is designed to address the vanishing gradient problem that can occur when training very deep neural networks. It adds a shortcut connection between the input and output of the block, which allows gradients to flow more easily during training. The block first applies a set of convolutional layers to the input, followed by an element-wise addition with the original input tensor. The result is then passed through an activation function such as ReLU[1].
In contrast, the inverted residual block starts with a depth-wise separable convolution. A linear projection then expands the number of channels, and the result is combined with the original input using a skip connection. Finally, a non-linear activation function such as ReLU, SiLU etc, is applied. Here we used ReLU6 as activation function. The "inverted" in the name refers to the fact that the Inverted Residual Block is designed to have a wider input than output. This is in contrast to traditional bottleneck layers, which have a wider output than input. By using the inverted design, the Inverted Residual Block reduces the number of computations required while maintaining high accuracy.
**Squeeze and Excite layer**: The Squeeze-and-Excite (SE) layer is a component of the MobileNetV3[2] architecture that improves the discriminative power of the network by selectively emphasizing informative features.
The SE layer consists of two main components: a squeeze operation and an excitation operation. The squeeze operation reduces the spatial dimensions of the input feature map by applying global average pooling(GAP). The resulting vector is then passed through two fully connected layers, which apply non-linear transformations to the features. The excitation operation then uses the learned weights to rescale the feature map channel-wise, producing an output with enhanced informative features.

The experiments carried out using the MobilenetV3 framework used a model that was similar to the one described in [2] with an adjustment in the final layer to accommodate four classes. With this modification, the model achieved acceptable accuracy in five epochs in a somewhat discontinuous fashion (refer Fig.(2)).

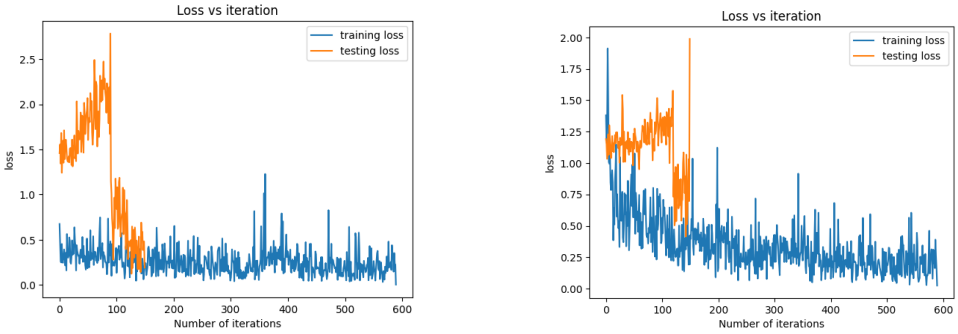


Figure 2: Loss trend for MobilenetV3(left) and MobilenetV3 with ECA layers(right)

### 4.3 Incorporating ECANet

This architecture focuses on avoiding dimensionality reduction and implementing effective cross-channel interaction[6]. The ECA layer consists of three main components: a global average pooling(GAP) layer, a convolutional layer and a sigmoid activation function.
In the ECA layer, GAP is applied before the 1D convolutional layer, which reduces the number of channels in the feature map by taking the average of each channel across all spatial dimensions of the feature map, producing a 1D vector with one value per channel.
The resultant vector is then passed through the 1D convolutional layer, which applies a weight to each channel based on its importance. The ECA layer replaces fixed size kernels of traditional convolutions with an adaptive kernel size.
The adaptive selection of kernel size in the ECA layer involves two main steps. First, the layer calculates the channel reduction ratio, which is a hyperparameter that controls the number of channels in the output feature map relative to the input feature map. Second, the layer uses the channel reduction ratio to determine the size of the kernel used in the 1D convolutional operation. Specifically, the kernel size is set to be equal to the number of input channels divided by the channel reduction ratio.
The resulting feature map is then passed through a sigmoid activation function, which produces a channel-wise attention map. Element-wise product of these channel-wise attention map with input tensor was done to rescale the feature map, producing an output with enhanced informative features. The ECA layer is usually placed at the last layer of residual or inverted residual block in CNNs. ECA maintains performance by avoiding complete independence among different groups with much lower model complexity.

For the current experiment, an ECA layer was added to every inverted residual block and then trained for five epochs; this did not significantly change the model's accuracy because an attention mechanism was already in place with the squeeze and excite layer. The loss trend(refer Fig.(2)) strongly resembled the loss trend of the usual MobilenetV3 model.

## 5 Plans

At present, the results have been quite promising; thus, it is expected that the current performance can be improved by extending the dataset and experimenting with hyperparameters like learning rate, momentum, k-size in case of an ECA layer to name a few. The possibility of improving the accuracy by using a more sophisticated model cannot be ruled out; hence, they will be explored in the future. Further, the performance of a non-neural network based model is yet to be gauged; this can indicate whether one should expect more performance from neural networks. Finally, a separate project based on [3], which uses GAN for generating patterns based on Indian classical sarees can be pursued.

## References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[2] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019.

[3] Harshil Jain, Rohit Patil, Utsav Jethva, Ronak Kaoshik, Shaurya Agarawal, Ritik Dutta, and Nipun Batra. Generative fashion for indian clothing. In *Proceedings of the 3rd ACM India Joint International Conference on Data Science &; Management of Data (8th ACM IKDD CODS &; 26th COMAD)*, CODS-COMAD '21, page 415, New York, NY, USA, 2021. Association for Computing Machinery.

[4] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2014.

[6] Qilong Wang, Banggu Wu, Pengfei Zhu, Peihua Li, Wangmeng Zuo, and Qinghua Hu. Eca-net: Efficient channel attention for deep convolutional neural networks. *CoRR*, abs/1910.03151, 2019.

[7] Zijie J. Wang, Robert Turko, Omar Shaikh, Haekyu Park, Nilaksh Das, Fred Hohman, Minsuk Kahng, and Duen Horng Chau. CNN explainer: Learning convolutional neural networks with interactive visualization. *CoRR*, abs/2004.15004, 2020.