ImageSplit: Modular Neural Network

Submitted by

M Elamparithy
(Reg No: 31020019)

In partial fulfillment of the requirements for the award of Master of Science in Computer Science with Specialization in Machine Intelligence

of

Cochin University of Science and Technology, Kochi

Conducted by

Indian Institute of Information Technology and Management-Kerala
Technopark Campus
Thiruvanathapuram-695 581

May 2022

BONAFIDE CERTIFICATE

This is to certify that the project report entitled ”ImageSplit: Modular neural Network” submitted by M Elamparithy (Reg No: 31020019) in partial fulfillment of the requirements for the award of Master of Science in Computer Science with Specialization in Machine Intelligence is a bonafide record of the work carried out at Indian Institute of Information Technology and Management under our supervision.

Supervisor	Course Coordinator
Alex P James	Asharaf S
Professor	Professor
IIITM-K	IIITM-K

DECLARATION

I, M Elamparithy,a student of Masters of Science in Computer Science with specialization in Machine Intelligence, hereby declare that this report is substantially the result of my own work , except where explicitly indicated in the text, and has been carried out during the period March 2022-June 2022.

Place:	Thiruvananthapuram
Date:	21-6-2022

ACKNOWLEDGEMENT

I express my deep sense of gratitude to Dr. Alex P. James, Associate Dean (Academic), Professor for his guidance and support all through the project. His comments and feedback were immense and apt to improve my understanding and explore the area. I also thank Dr. Chitra R, for her guidance and contribution. I express my gratitude to the course coordinator, Dr. Asharaf S for his guidance. I also thank my parents and friends for their support.

ABSTRACT

Neural network architectures are computationally intensive.They end up taking up hours of training time despite being trained on GPUs.With time, the sizes and complexity of neural network architectures being implemented is growing, leading to high power consumption.Alternative architectures are a need of the hour.Techniques that take less time to train and can also perform as good as the state of the art are very valuable.Image classification, a well known and fundamental problem in the field of computer vision is usually done using huge neural network architectures.In this report we explore alternative neural network architectures that minimize computational complexity and test their performance in an image classification task.We propose a new technique called "ImageSplit" to perform image classification which is done by splitting an image into non overlapping blocks and processing each individual block separately. The split blocks of the image are processed independently. Classification is done by combining the class prediction of each individual block.This approach is experimented on three different datasets with various neural network architectures.The results obtained show that splitting and processing an image gives a much better accuracy as compared to processing the image as a whole.

Table of Contents

List of Tables vii

List of Figures viii

1 Introduction 1
- 1.1 Deep Learning . . . 1
- 1.2 Image Classification . . . 2
- 1.3 Complexity: a challenge in deep learning . . . 5
- 1.4 Introducing ImageSplit . . . 6
- 1.5 Contribution . . . 7

2 Literature Review 8
- 2.1 Complexity Reduction Techniques . . . 8
- 2.2 Exploring Splitting techniques . . . 9

3 Materials and Methods 15
- 3.1 ImageSplit methodology . . . 15
  - 3.1.1 Concept . . . 15
  - 3.1.2 Derivation . . . 19
- 3.2 Architectures . . . 20
  - 3.2.1 Single layered Neural Networks . . . 20
  - 3.2.2 Multiple layered Neural Networks . . . 20
  - 3.2.3 Simple Convolutional Neural Networks . . . 21
  - 3.2.4 Pre-trained Models: ResNet50, VGG16 . . . 21

4 Result and Evaluation 25
- 4.1 Datasets Used . . . 25
- 4.2 Experiments and Results . . . 27
  - 4.2.1 Single hidden layer . . . 27
  - 4.2.2 Multiple hidden layers . . . 28
  - 4.2.3 Convolutions . . . 28

4.2.4 Pre-trained models 29
4.3 Observations 31

5 Conclusion **33**

REFERENCES **34**

List of Tables

4.1 Single Hidden Layer 28
4.2 Multiple hidden layer 28
4.3 Convolutional Neural Network 29
4.4 VGG16 30
4.5 ResNet50 31

List of Figures

1.1 Image classification of Cat and Dogs 2
1.2 Simple Artificial Neural Network 4
1.3 Simple Convolutional Neural Network 5

2.1 Splitiing convolutional layer 10
2.2 SplitNet 11
2.3 Split-CNN 12
2.4 Slim-CNN 13

3.1 Single vs divided input 16
3.2 Splitting 16
3.3 BAGGing: Dataset Sampling 17
3.4 Splitting image 17
3.5 3x3 and 2x3 split 18
3.6 Splitting and processing: the whole process 18
3.7 Architecture: Simple CNN 21
3.8 VGG16 Architecture 23
3.9 ResNet50 Architecture 23

4.1 Intel Image Classification 26
4.2 STL-10 26
4.3 CIFAR-10 27
4.4 Single hidden Layer graph 29
4.5 VGG16 performance 30
4.6 ResNet50 graph 31

Chapter 1

Introduction

1.1 Deep Learning

Within the area of Machine Learning, deep learning is a subfield which uses neural networks to model high level abstractions in data.According to [1], deep learning methods learn representations.Beginning with the input, each non-linear module in the deep learning algorithm transforms the representation at one level to a slightly higher level in abstraction.A comprehensive representation is obtained after the input goes through several non-linear transformations.Deep learning methods perform well in exploring high-dimensional data as compared to the conventional machine learning algorithms.Deep learning has performed successfully in fields [1] like computer vision and natural language processing [2, 9].A big factor for this success is due to availability of relatively cheap computational power.
Convolution neural networks, a particular type of deep learning algorithm, had an exceptional performance in the ImageNet competition (2012) [2].It had half of the error rates of the best competing approaches.CNNs have become the most frequently used approach for all image recognition and object detection tasks in the computer vision community ever since.Image processing in CNNs is inspired from how humans see and process the world around them through their vision.In CNNs, the data processing is done in the form of multiple arrays.According [1], CNN methods can have upto hundreds of millions of weights and billions of connections between individual units for a model that has 10-20 layers.Models like these, with their algorithm parallelization and modern hardware can take up several hours to train [1].Recurrent neural networks (RNNs) have performed well for the tasks that deal with speech and text, i.e data which is sequential in nature.RNNs are

designed to predict the next character or word given a sentence's context.[3] used two multiple layered LSTM RNNs to translate sentences in English to French.[4] uses long short-term memory (LSTM) RNNs to perform speech recognition, which achieves a test set error of 17.7% on the benchmark testing data.

1.2 Image Classification

Image classification is a typical machine learning task where Deep Learning architectures have performed very well.The main goal of the classification process is to categorize the objects in a digital image into one of several classes as shown in Figure 1.1 .In other words, image classification involves finding an association between the features occurring in an image and the object these features actually represent on the ground.Usually, multi-spectral data are used to perform the classification.The value of each pixel and how it interacts with other pixels to form a bigger picture is used as the numerical basis for categorization of the data.Image classification is used very frequently in digital image analysis.

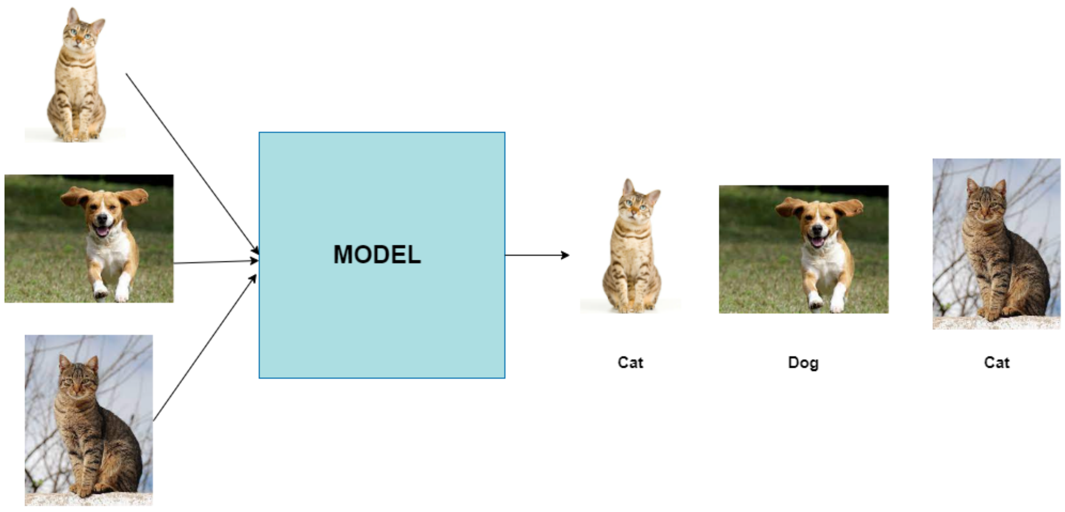


Figure 1.1: Image classification of Cat and Dogs

Classifying different objects is very difficult and hence image classification has been an important task in the computer vision community.A machine learning algorithm performs Image classification by labelling images into one of a number of classes which are predefined.There can be any number of classes to which a given image belongs to.It is impractical to manually check and classify images when there is a huge dataset of images.Hence it will be very useful if automating this process using techniques in computer vision and machine learning algorithms.Image classification has played a huge part in the advancement of the field of autonomous driving.Other applications of image classification include automated image organization, image and face identification on social networks, visual databases, stock photography and websites using video features, visual search for better product searchability and many more which is why we need classifiers with good performance.

Structure for performing Image Classification

1. Pre-processing: This is done to improve the image data (the relevant features) by removing unwanted distortions and enhancing a few crucial image features so that the computer vision models can work on this improved images more efficiently.Some of the image pre-processing techniques include- reading and resizing the image, data augmentation which is done to create more images from the existing image dataset.Data Augmentation involves - Gray scaling of image, Reflection, Gaussian Blurring, Histogram, Equalization, Rotation, and Translation.

2. Localizing the object: Segmenting various parts of the image and identifying the position of the object which needs to be classified.

3. Extracting features and training: It is the most important step as deep learning and statistical methods are used to identify the most interesting patterns in the image, features and characteristics that might be specific to a particular class and that will be useful for the model to differentiate between different classes.This is the most crucial step in image classification and is also referred to as Model training, as the model is learning important features.

4. Classification of the object: Categorizing objects detected in the image into suitable classes by using a classification technique mentioned in the previous step that compares the features extracted from the image with the target patterns.

Artificial Neural Network

Artificial Neural Networks are statistical learning algorithms inspired from the properties of biological neural networks.They are used for a variety of tasks, from comparatively simple classification tasks to speech recognition and computer vision.The most fundamental unit of an artificial neural network is the node or neuron.As shown in Figure 1.2, a neural network is nothing but an interconnected system of nodes.These nodes model the functionality of biological neurons.The connections between different nodes are called weights which have different numerical values.Changing these values in a systematic and iterative way leads the neural network to approximate the desired function.The hidden layers detect individual features in the image.As it is propagated throughout the network, the layers recognize more and more complex patterns in the image.For example, if the network's task is to detect an object, the initial hidden layers might act as a edge detector, subsequent layers will take these edges as input and combine them together to form a line, the final layers take the lines and matches it with an boundary and so on, until finally the whole object boundary is detected.The network can detect very complex objects using this hierarchy.There are different types of artificial neural network, namely feedforward neural network, convolutional neural network, recurrent neural network, time delay neural network, probabilistic neural network, deep stacking network and radial basis function network.

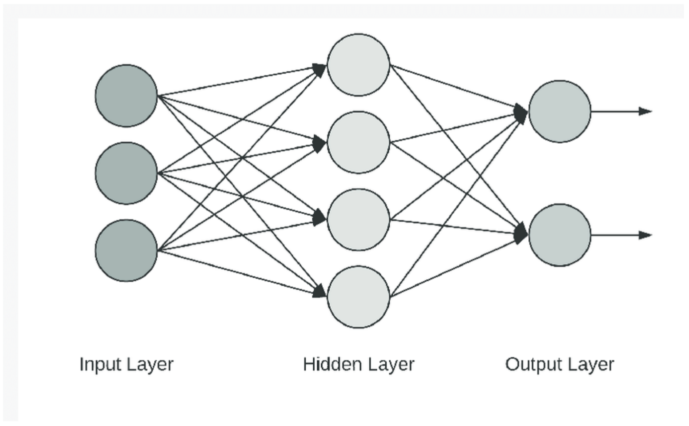


Figure 1.2: Simple Artificial Neural Network

Convolutional Neural Network

Convolutional Neural Network, also known as ConvNet or CNN are a type of multi-layered neural networks which are designed to identify visual patterns in images with negligible pre-processing.It is different from regular artificial neural networks as it uses an additional layer called convolutional layer.CNNs are inspired inspired from the functioning of the human visual cortex and have achieved state of the art results in computer vision tasks.Convolutional neural networks have two simple elements- convolutional layers and pooling layers.Although, these elements are simple and easy to understand individually, the hard part of implementing a convolutional neural network in practice is designing the model architectures that use these simple elements efficiently.For a given computer vision problem, there are countless ways to arrange these layers to create an efficient model.CNNs are hugely popular because of their architecture as convolutional layers take care of feature extraction which is the most difficult part in any computer vision task.The core concept of a CNN is that it uses convolutions and filters on an image to generate invariant features as shown in Figure 1.3.These features are passed onto the next layer.The features in the next layer are convoluted with different filters to generate more abstract features and the process continues till it gets a final feature or output which is invariant to occlusions.The most frequently used architectures of convolutional neural network are VGGNet [33], AlexNet, GoogLeNet, LeNet, ResNet [34] and ZFNet.

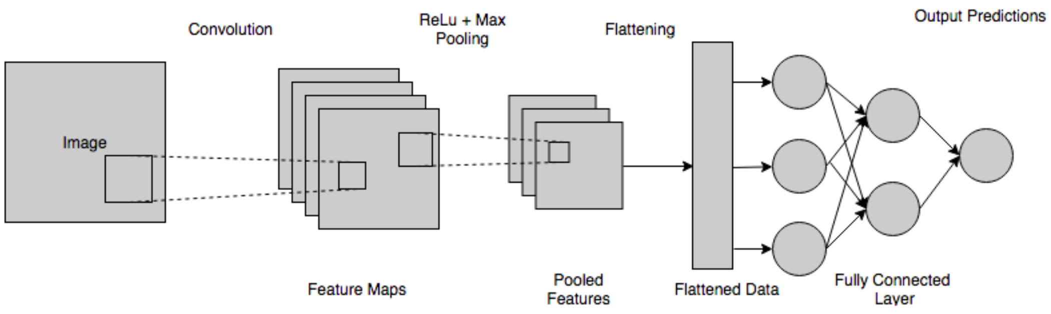


Figure 1.3: Simple Convolutional Neural Network

1.3 Complexity: a challenge in deep learning

Deep neural networks have performed extremely well on a lot of tasks, including visual recognition [2], speech recognition [5], and natural language processing.However, such impressive performances are achieved at the cost of increased computational resources compared to traditional machine learning models including shallow neural networks.Training and testing of deep learning algorithms are very computationally intensive.High complexity of deep networks can cause problems if the model and the task size becomes very large(e.g. Detecting thousands of different objects), or the application

is time-critical (e.g. real-time face verification).
There are multiple approaches to tackle this computational need of deep learning algorithms.One way is to reduce the number of model parameters; this could be achieved in several ways-

1. Training a new smaller network while maintaining similar behavior as the original network [6].

2. Deleting unnecessary weights through pruning or sparsity regularization [7].

Another approach to speed up deep networks is distributed machine learning [8], however most research effort has been made on the systems and optimization sides, without consideration of the ways to obtain network structure that is intrinsically scalable.

1.4 Introducing ImageSplit

Deep learning architectures are computationally intensive. Is it necessary for a data point in the dataset to be processed as a whole? The intensive part in deep learning networks is the matrix multiplication of millions of parameters. Are all these computations necessary? Can we afford to break the data in small parts at the risk of losing accuracy but gaining speed? This project report describes experiments with images where, instead of processing them as a whole, they are split into multiple blocks and processed by smaller networks.
We call this novel idea "ImageSplit", to reduce the computational complexity of DNN implementation using the idea of image splitting. In ImageSplit, the input image in the dataset is split into smaller units and each units are processed using small sized neural networks.Multiple smaller networks instead of one big network can save computations and be much faster if run in parallel.The performance is evaluated for different datasets including the Intel Image Classification, STL-10 and CIFAR-10 data-sets.

1.5 Contribution

The concept and theory of ImageSplit will be explained in detail in the upcoming sections.The primary contributions of this report are-

1. A novel method of image classification wherein the image is split into smaller blocks and processed separately. This is discussed in detail in *section 3.1.1*
2. A theoretical insight on how ImageSplits reduces number of computations performed during processing. *(section 3.1.2)*
3. Experimental results of the performance of ImageSplit methodology on multiple image datasets. *(section 4.2)*

Chapter 2

Literature Review

2.1 Complexity Reduction Techniques

Achieving test-time efficiency and parameter reduction in deep neural networks is an active research topic in deep learning.A simple approach is to apply the ‘1-norm [10] which removes weak connections during the training. However, the ‘1- norm often results in a model that trades-off the accuracy with the efficiency.[11] presented an iterative weight pruning technique that repeatedly retrains the network while removing all the weak connections.This technique achieves a superior performance over ‘1-regularization.Recently, the group sparsity using ‘2,1-norm has been explored for learning a compact model.[12] applies (2,1)-norm regularization in every single layer to eliminate the hidden units that are not shared across upper-level units.This automatically decides how many neurons are required at each layer. [13] used the same group sparsity to select irrelevant channels and spatial features in a CNN, and let the network to decide automatically on how many layers to use.However, they assume that all classes share the same set of features, which is restrictive when the number of classes is very large.Recently, [15] also addressed the use of symmetrical split at mid-level convolutional layers in CNNs for architecture refinement.

Another approach is parallel and distributed deep learning.As deep networks and training data become increasingly larger, researchers are exploring parallelization and distribution techniques to speed up the training process.Most parallelization techniques exploit either 1) data parallelism, where the training data is distributed across multiple computational nodes, or 2) model parallelism, where the model parameters are distributed.[14] used both data and model parallelism to train a large-scale deep neural network on a computing cluster with thousands of machines.For CNNs, [2] used both data and model parallelism to train separate convolutional filters from disjoint datasets. [16] later proposed to splitting the network vertically by which different time/memory characteristics of the convolutional and fully connected layers could be exploited.[17] proposed a server-client architecture where each client computes partial gradients of parameters, that are stored and communicated from the global model parameter server.[18] proposes a GPU-based distributed deep learning, which significantly reduces the communication overheads.

2.2 Exploring Splitting techniques

There are several existing techniques to reduce the computational complexity of deep neural network architecture [19, 20, 22, 23].These involve splitting the network structure into small parts which can consume lower memory than the original network.

Splitting the convolutional layer

In Figure 2.1, the proposed split model for CNN is depicted.A fusion part is designed after the splitting to preserve global information and provide a suitable input size to the next split block.The fusion block constitutes of three steps including pooling operation, concatenation, and 1×1 convolution. The agorithm used to obtain the model in Figure 2.1 takes the following steps-

1. At first, a baseline network is trained to reach its final accuracy.

2. After that, some blocks are considered for splitting. For better consistency and more reduction in the number of feature maps, the fusion block can be located on the location of pooling operations in the original network.

3. The number of split blocks could be the same as the number of pooling operations.The first feature maps in the CNNs have a larger size than the final ones.Therefore, the splitting process starts from the initial layers by a splitting factor of two.

4. After splitting, the model is fine-tuned in a few epochs, and its accuracy is evaluated.In each block, splitting is continued until a higher than a given threshold of accuracy is observed.

In this way, each block can be split based on its capacity for splitting.Based on the proposed method, some layers may not be split at all due to their importance and deep feature map dependencies.

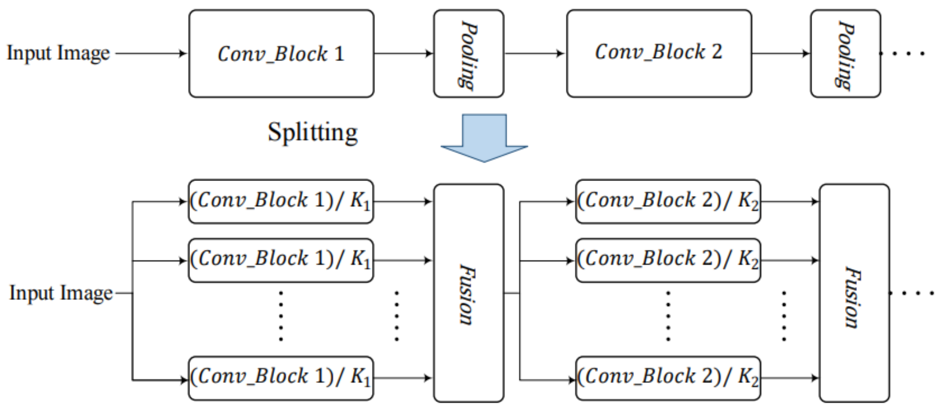


Figure 2.1: Splitting the convolutional layer, Reprinted from [19]

SplitNet

SplitNet in [20] splits the network weights into either a set or a hierarchy of multiple groups that use disjoint sets of features, by learning both the class-to-group and feature-to-group assignment matrices along with the network weights.Thus, it reduces the number of parameters and required computations.Its goal is to train a deep neural network that not only minimises the amount of model parameters but is also well-structured for parallelization.This is accomplished by emphasising the idea that as the number of classes grows, semantically distinct classes (or tasks) can be represented by largely independent sets of characteristics.For example, features describing cars may be quite different from those describing hills, whereas low-level features such as stripes, dots, and colors are likely be shared across all classes.It implies that depending on the traits they utilize, we can arrange classes into mutually exclusive groupings.Such grouping of concepts based on semantic proximity also agrees with the way that our brainstores semantic concepts, where semantically related concepts activate similar part of the brain [21], in a highly localized manner.Based on this observation SplitNet automatically performs deep splitting of the network into a set of subnetworks or a hierarchy of subnetworks sharing common lower layers, such that classes in each group share a common subset of traits that are fully distinct from those in other class groupings We train such a network by learning classes-to-group assignments and feature-to-group assignments in addition to the network weights, and then disjointing them across groups.Figure 2.2 illustrates the key idea of SplitNet.

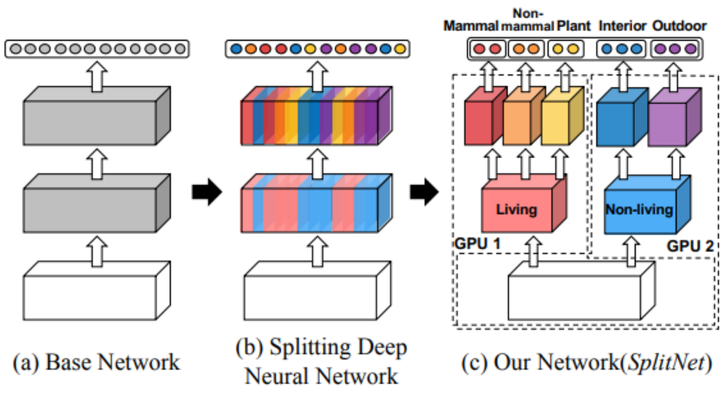


Figure 2.2: The network automatically learns to split the classes and associated features into multiple groups at multiple network layers, obtaining a tree-structured network. Reprinted from [20]

Split-CNN

Split-CNN [22] identifies the frequency of memory bound layers, increasing batch sizes, and increasing model complexity as the three challenging trends leading to the memory bottleneck of training deep neural networks.It elaborates on the cause and impact of these trends on DNN training systems and tackles the three challenges, by proposing the following ideas-

1. Firstly, Split-CNN, a general instrumentation of conventional CNN models that separates memory bottlenecks while keeping or even improving the model's application-specific metrics (i.e., classification accuracy).Split-CNN was assessed and its impact on application metrics was quantified across a variety of model and dataset combinations.

2. Second, a new heterogeneous memory management system (HMMS) that allows memory allocation, de-allocation, offloading, and prefetching to be scheduled optimally without any network model modification.The evaluation results for its offloading/prefetch plan are then shown to illustrate its ability to solve the challenge with memory-bound layers.

3. Thirdly, combining Split-CNN and HMMS to showcase that Split-CNN can be trained with significantly larger batch size, thus effectively solving the challenges with the increasing demands on large batch size.

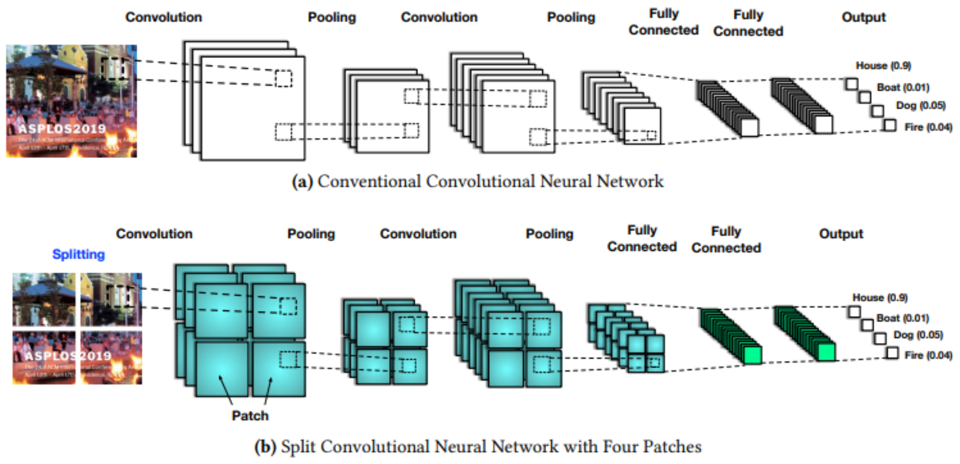


Figure 2.3: Regular CNN vs Split-CNN Comparison. Reprinted from [22]

Experiments show that Split-CNN significantly increases the scalability of the training algorithm and the HMMS can schedule memory allocations and host-device data movements optimally with no tuning required.Combining these two techniques has shown that VGG-19 models and ResNet-18 mdodels can be trained with 6 times and 2 times larger batch size respectively compared to the baseline method.Furthermore, since Split-CNN enables model training with large batch sizes, it can improve the performance of the distributed training by reducing the network traffics caused by parameter updates.Split-CNN is capable of achieving 2.1x speedup in the distributed training for VGG-19 model.

Slim-CNN

Slim-CNN's [23] goal is to provide a high-performing, computationally efficient CNN architecture with fewer parameters than previous models.To achieve this purpose, it uses CNN micro-architectures, which are separate modules made up of convolutional, pooling, fully-connected, and dropout layers.Micro-architectures such as the Inception module and Residual blocks have been demonstrated to provide good discriminative feature representations.The innovative micro-architecture called the 'Slim Module' is made up of two types of convolutional layers (depth separable convolution for 3x3 kernels and pointwise convolution for 1x1 kernels), as well as skip-connections.Figure 2.4 shows the architecture of a Slim module, it consists of a SSE block which is a multi-layered arrangement consisting of two 1x1 pointwise convolutional layers and a single 3x3 depthwise separable convolutional layer.

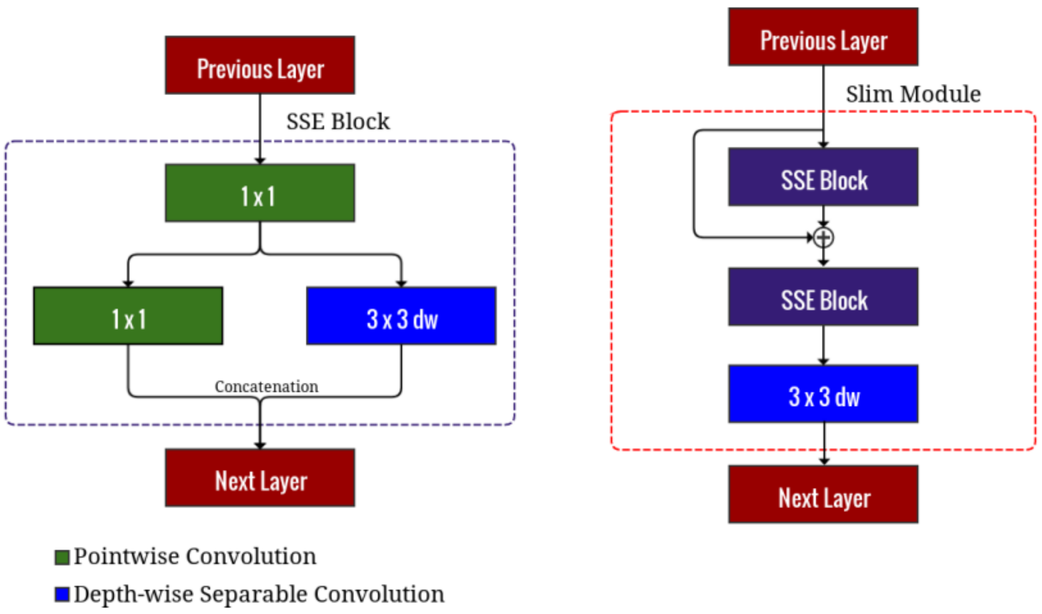


Figure 2.4: Slim Module: The micro-architecture shown on the left is the Separable SqueezeExpand Block (SSE), while on the right is the Slim Module. Reprinted from [23]

The main contributions of [23] are-

1. It presents a new CNN Slim Module shown in Fig. 2.4 that is compact, computationally efficient, has a smaller memory footprint, and still produces excellent discriminative features.
2. It investigates and compares the proposed Slim module micro-architecture with other micro-architectures in the literature for various kernel sizes.
3. It evaluates the performance of a stacked Slim Module deep neural network Slim-CNN in terms of accuracy, parameter size, and on-disk memory space with state-of-the-art approaches on face attribute prediction and shows significant improvement.

The works in [19, 20, 22, 23] discusses network complexity reduction by splitting the neural network itself.The computing complexity reduction by image splitting is not discussed in the literature and is a new direction.In the next chapter we explore this concept image splitting by explaining in depth the philosophy of 'ImageSplit' with several illustrations and theoretical insights.

Chapter 3

Materials and Methods

3.1 ImageSplit methodology

We now introduce ImageSplit, a novel idea for parameter reduction through the concept of splitting-

3.1.1 Concept

Image processing in a neural network is a series of matrix multiplications where we take as input the individual pixels or the feature maps of the image depending on the network architecture.Since matrix multiplications grow in the order of $O(n^3)$, we explore whether every multiplication in the matrix between two layers is necessary. This is done by splitting the image matrix into smaller non overlapping sub matrices.

As shown in Figure 3.1 the number of parameters reduce significantly when taking a divided input as compared to the whole input.Hence, the total number of computations is reduced since the weight multiplication for the single matrix is more than multiple smaller matrices.Although the number of computations decreases, there are multiple outputs since the image is split and processed by smaller networks, so the final output is obtained by taking a vote.This process is performed on architectures of various complexities.Convolutions on an image produce feature maps which are further convolved to obtain more meaningful features.By performing convolutions on smaller split images, the network is being trained to recognize a particular part of the whole image and its relationship with the class of the object in the image.After training, there are multiple smaller networks which are trained

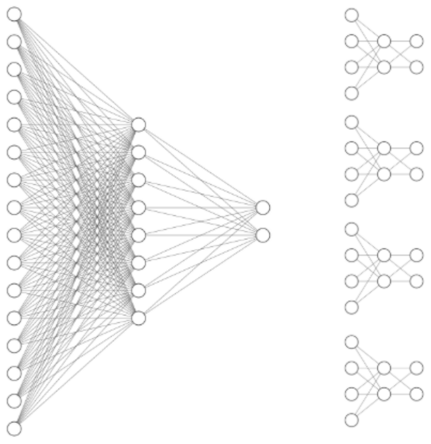


Figure 3.1: Single vs divided input

to predict the class of the image by just examining a single part of the image.Since there are multiple predictions, a voting is taken to decide the final prediction, so the prediction is what the majority of the smaller networks propose the class as.Images in the dataset are split as shown in Figure 3.2, this in particular, is the 2x2 split. Other splits like 3x3 and 2x3 are also used.More splits require fewer computations to be performed per split but the number of splits increase.This paper tackles the question of whether this process of splitting is feasible and the model obtained has a decent accuracy.

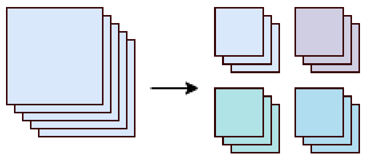


Figure 3.2: Splitting

This process is applied to the single layered neural network, network with multiple layers,a simple convolutional neural network and complex architectures like VGG16 [33] and ResNet50 [34].In the case of ResNet50 and VGG16 pre-trained models are used since they are already good at classifying everyday objects.

The method used is similar to Ensemble learning algorithms [24].BAGGing [25] is an ensemble learning algorithm where, given a sample of data, multiple bootstrapped subsamples are pulled as shown in Figure 3.3.A model is formed on each of the bootstrapped subsamples.After each subsample model has been formed, an algorithm is used to aggregate over the models to form the most efficient predictor.

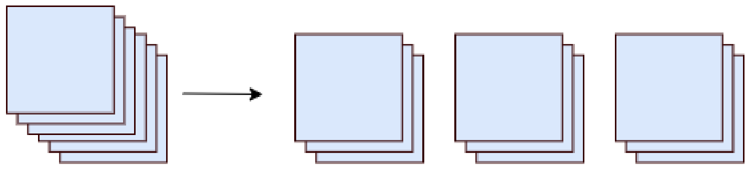


Figure 3.3: BAGGing: Dataset Sampling

Bootstrapping: Bagging leverages a bootstrapping sampling technique to create diverse samples.This resampling method generates different subsets of the training dataset by selecting data points at random and with replacement.This means that each time you select a data point from the training dataset, you are able to select the same instance multiple times.As a result, a value/instance repeated twice (or more) in a sample. Parallel training: These bootstrap samples are then trained independently and in parallel with each other using weak or base learners. // Aggregation: Finally, depending on the task (i.e. regression or classification), an average or a majority of the predictions are taken to compute a more accurate estimate.In the case of regression, an average is taken of all the outputs predicted by the individual classifiers; this is known as soft voting.For classification problems, the class with the highest majority of votes is accepted; this is known as hard voting or majority voting.

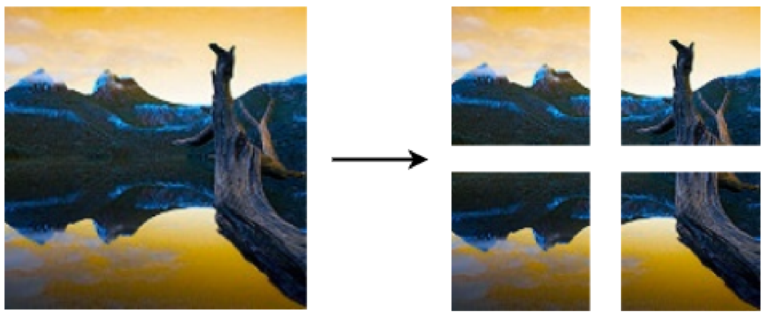


Figure 3.4: Splitting image

The difference here is that in case of splitting, multiple datasets are formed by dividing every single data point itself (Figure 3.2) into multiple parts as opposed to taking bootstrapped subsamples from the dataset.Random Forest [26] is an ensemble learning algorithm which is even more similar to splitting.Similar to BAGGing, bootstrapped subsamples are pulled from a larger dataset.A model is formed on each subsample.However, the model is split on different features.

The method involves splitting each image in the dataset into multiple non-overlapping blocks.These blocks can be of various sizes.In a 2x2 split, the image is divided into 4 blocks as shown in Figure 3.4.We primarily experiment with experiment with the 2x2, 3x3 and 2x3 splits shown in Figure 3.5.

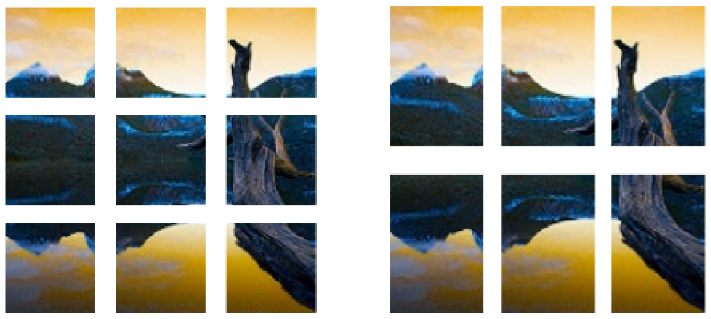
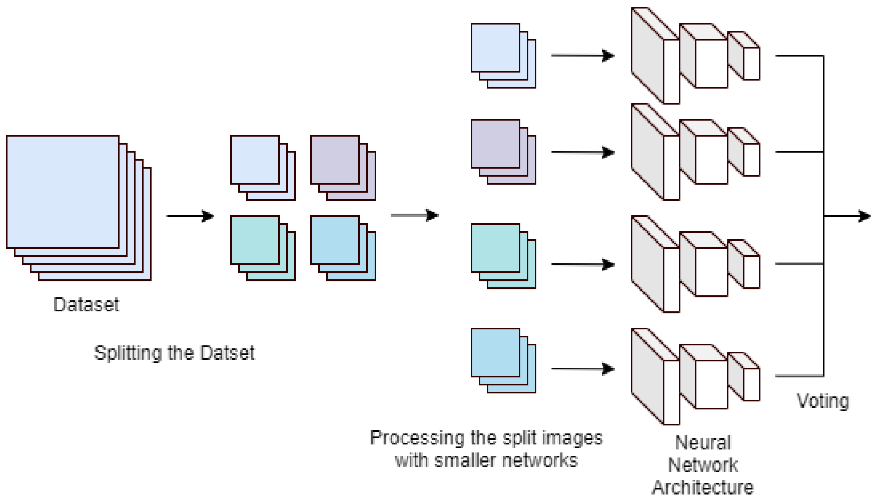


Figure 3.5: 3x3 and 2x3 split

After splitting, these images are separated into their own datasets, i.e for a 2x2 split, we will have four smaller datasets, one dataset containing the upper half, one containing the lower half of all images and so on as shown in Figure 3.2.These smaller datasets are fed into the smaller neural network architectures as shown in Figure 3.6


Dataset
Splitting the Datset
Processing the split images with smaller networks
Neural Network Architecture
Voting
Figure 3.6: Block diagram of the Proposed ImageSplit Methodology (a) ImageSplit:Image split into datasets, (b) Processing:Images fed into multiple smaller neural network architectures, (c) Multiple neural network Architectures for processing (d) Combining: Among multiple classes, the class which is deteced by most of the networks is chosen as the final output.

3.1.2 Derivation

Image processing in a neural network is a series of matrix multiplications where we take as input the individual pixels or the feature maps of the image depending on the network architecture.Since matrix multiplications grow in the order of $O(n^3)$, we explore whether every multiplication in the matrix between two layers is necessary.This is done by splitting the image matrix into smaller non overlapping sub matrices.Multiple smaller matrix multiplications will be smaller than a single bigger matrix multiplication.A forward pass in a neural network is the matrix product between the input vector and weight matrix.Consider a matrix multiplication between a vector, $V = [x_1, x_2, ..., x_n]$ and a weight matrix, W. Multiplying V and W will take $a \times b$ operations.Now consider splitting V into L subvectors parts, $V = \{V_1, V_2, ..., V_L\}$,where $V_i = \left\{x_{\frac{(i-1)n}{L}+1}, x_{\frac{(i)n}{L}+2}, ..., x_{\frac{(i)n}{L}}\right\}, i \in \{1, L\}$. Now, these $V_i's$ are individually multiplied with smaller weight matrices W_i with dimension $(a_1 \times b_1)$, $a_1 < a, b_1 < b$.

Figure 3.6 shows the proposed ImageSplit methodology.Consider a dataset with K images each with size $a \times b$.In splitting, each K^{th} image is split into L sub-units with size $a1 \times b1$.In Figure 3.6, L, $a1 \in \{1, 2...a\}$ and $b1 \in \{1, 2...b\}$.Figure 3.4 shows how the original can be divided into L splits with multiple non-overlapping blocks.These blocks can be of various sizes, 2×2, 3×3 and 2×3 splits are shown in Figure 3.5.After splitting, these images are separated into their own datasets, i.e for a 2×2 split, we will have four smaller datasets, one dataset containing the upper-right half, one containing the lower-right half of all images and so on.

The L separate blocks from the image split are processed separately using smaller neural network architectures.Each L splits of size $a1 \times b1$ forms input to L parallel neural network architectures.The smaller individual networks process the data and classify it.For a dataset with n number of classes, each neural network has n-tuple as its output, $X_1 = \{x_{m,1}, x_{m,2}...x_{m,n}\}$, where $m \in \{1, 2, ..L\}$.For example, $L = 4$ split has four outputs in total $[X_1\ X_2\ X_3\ X_4]$.Although the number of computations decreases, there are multiple outputs since the image is split and processed by smaller networks.For each k_{th} image, the final output is computed by taking the four outputs detected from each network.These four outputs may be same or different.The classes detected are counted in Figure 3.6.The final output for k_{th}

image computed from $[X_1\ X_2\ X_3\ X_4]$ is as:

$$\{y_{k,n}\} = \left\{ \sum_m x_{m,n} \right\} \forall\ n \tag{3.1}$$

$\{y_{k,1}, y_{k,2}...y_{k,n}\}$ are computed from Eq (3.1).The class that is detected most often by the smaller networks is chosen as the final prediction.The predicted output for k_{th} image is

$$y_{k,i} \leftarrow \arg\max_n \{y_{k,n}\}, 1 \leq i \leq n \tag{3.2}$$

Eq. 3.2 implies that class $y_{k,i}$ has the highest number of counts *i.e*, it has been detected the most often and is chosen as the final output class.

3.2 Architectures

We experiment with different types of Neural network architectures which get complex progressively as we add more and different types of layers.The architectures used are-

3.2.1 Single layered Neural Networks

Since they are easy to implement, numerous types of splits are is experimented with- 2x2, 3x3, 5x5, 6x5, 6x6, 10x5, 10x6, 15x5, 10x9, 10x10.The single layer has as many neurons as the input dimension of the flattened image.

3.2.2 Multiple layered Neural Networks

Multilayered networks with four layers.The first layer has number of neurons equal to the dimensions of the input flattened, and three fourth, half, one fourth of the input in the subsequent layers.
Convolutional Neural Networks are an addition, improvement to the traditional neural.With the addition of the convolutional layer, the CNN is able to extract relevant features which are then processed by the neural network/-fully connected layer.

3.2.3 Simple Convolutional Neural Networks

Three convolutional layers followed by two dense layers.The first convolutional layer has a kernel size equal to the dimensions of the image.The subsequent layers have a size of half and one fourth the input dimensions.Figure 3.7 shows the architecture of a single network for the 5x5 split.

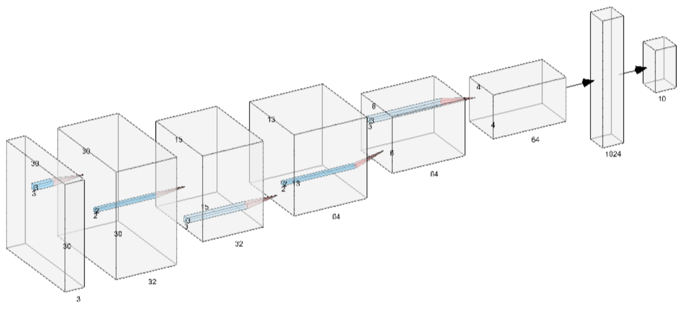


Figure 3.7: Architecture: Simple CNN

3.2.4 Pre-trained Models: ResNet50, VGG16

We introduce the concept of transfer learning before we talk about ResNet50 and VGG16 architectures -

Transfer Learning

During the development phase, supervised machine learning models are trained to accomplish specific tasks using labelled data.The algorithm is given with input and output that are clearly mapped.The model can then use the learned patterns and trends to recognise fresh data.When completing tasks in the same environment as their training data, models created this way will be extremely accurate.If the conditions or surroundings change in real-world application beyond the training data, it will become significantly less accurate.Even though the tasks are similar, a new model based on new training data may be required.Transfer learning is a method for addressing this issue.It works as a concept by transferring as much information as feasible from an existing model to a new model meant to do the same task.Transferring the more generic components of a model that make up the main processes for completing a task, for example.This could be the procedure for identifying and categorizing items or photos.The new model can then be enhanced with additional layers of more particular information, allowing it to fulfil its duty

in new contexts.
Transfer learning [32] entails taking the relevant portions of a machine learning model that has already been trained and applying them to a new but related problem.This will often comprise the model's core information, with new characteristics added to the model to address a given problem.Programmers must determine which aspects of the model are important to the new task and which elements must be retrained.A new model, for example, might preserve the mechanisms that allow the computer to recognise objects or data but retrain it to recognise a different specific object.A machine learning model which identifies a certain subject within a set of images is an ideal candidate for transfer learning.The majority of the model, which focuses on how to recognise various subjects, can be preserved.The element that will be retrained is the part of the algorithm that identifies a certain subject to categorise.There's no need to rebuild and retrain a machine learning system from the ground up in this situation.

VGG16

VGG16 [33] is a CNN (Convolutional Neural Network) that is widely regarded as one of the best computer vision models available today.It's a well known CNN architecture that's simple to utilise with transfer learning.VGG16 is a 92.7 percent accurate object recognition and classification system that can classify images into 1000 different categories.

1. The 16 in VGG16 stands for 16 weighted layers. VGG16 comprises thirteen convolutional layers, five Max Pooling layers, and three Dense layers, for a total of twenty-one layers, but only sixteen weight layers, or learnable parameters layers.

2. The input tensor size for VGG16 is (224, 244) with three RGB channels.

3. The most distinctive feature of VGG16 is that, rather than having a huge number of hyper-parameters, they focused on having 3x3 filter convolution layers with stride 1 and always used the same padding and maxpool layer of 2x2 filter stride 2.

4. The convolution and maximum pool layers are placed in a regular pattern throughout the architecture.

5. The Conv-1 Layer has 64 filters, the Conv-2 Layer has 128 filters, the Conv-3 Layer has 256 filters, and the Conv 4 and Conv 5 Layers have 512 filters.

6. Following a stack of convolutional layers, three Fully-Connected (FC) layers are added: the first two have 4096 channels apiece, while the third performs 1000-way ILSVRC classification and so has 1000 channels (one for each class). The final layer is the soft-max layer.

Figure 3.8 shows the architecture described.

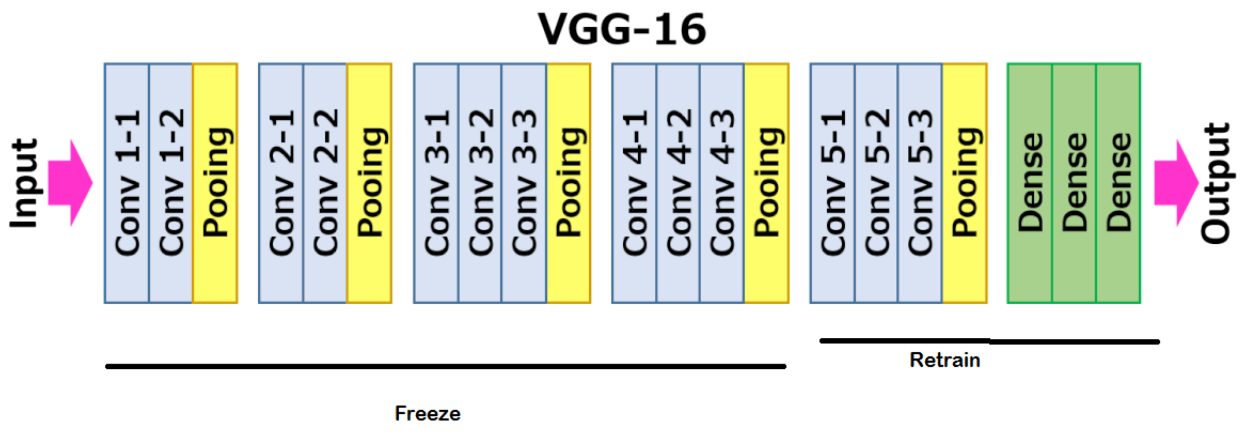


Figure 3.8: VGG16 Architecture, Reprinted from [27]

ResNet50

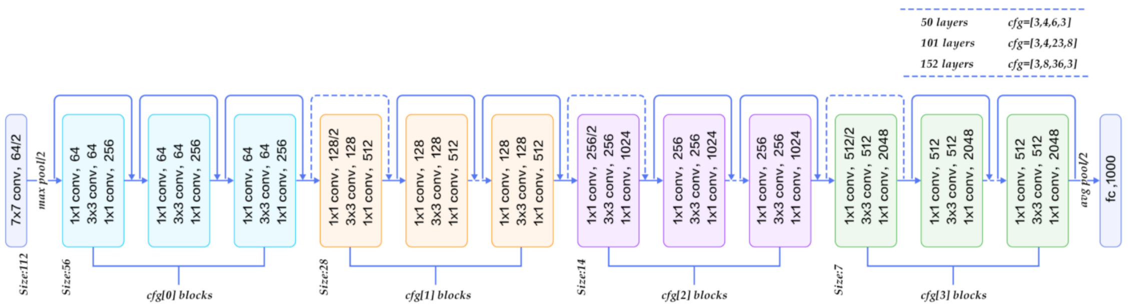


Figure 3.9: ResNet50 Architecture, Reprinted from [28]

ResNet-50 [34] is a convolutional neural network that is 50 layers deep and has over 23 million trainable parameters.The ResNet50 model consists of 5 stages each with a convolution and Identity block as shown in Figure 3.9.Each convolution block has 3 convolution layers and each identity block also has

3 convolution layers.ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks.The fundamental breakthrough with ResNet was the concept of 'skip connection' which enables training extremely deep neural networks with more than 150 layers.
Since these models are very complex, we perform transfer learning [32] to avoid training all the parameters.Both these models are only used as feature extractors since they are good at classification tasks.Separate trainable fully connected layers are added on top of the pre-trained model.
The smaller individual networks process the data and classify it.For a 2x2 split, there are four predictions, one from each network.These four predictions can be different, so a voting is taken to decide the class of the image as shown in Figure 3.6

Chapter 4

Result and Evaluation

We detail the experimental results and observations performed using the ImageSplit method.Firstly, we discuss the datasets that are used to evaluate the performance of the method.

4.1 Datasets Used

The ImageSplit methodology is evaluated on the following three datasets-

Intel Image Classification

This dataset [30] contains around 25k images of size 150x150 distributed under 6 categories.As shown in Figure 4.1, the categories are:Building, Forest, Glacier, Mountain, Sea, Street.

STL-10

The STL-10 dataset has 10 classes: airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck, Figure 4.2 With color images of size 96x96.The dataset has 500 training images 800 test images per class.It is inspired by the CIFAR-10 dataset but with some modifications.The images in this dataset were acquired from labeled examples on ImageNet.

CIFAR-10

.

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class.The classes are: airplane, automobile, bird, cat,

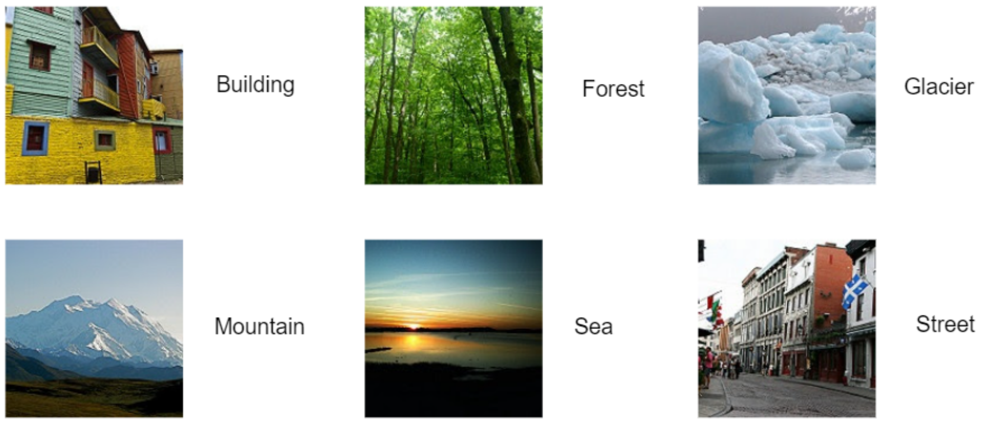


Figure 4.1: Intel Image Classification

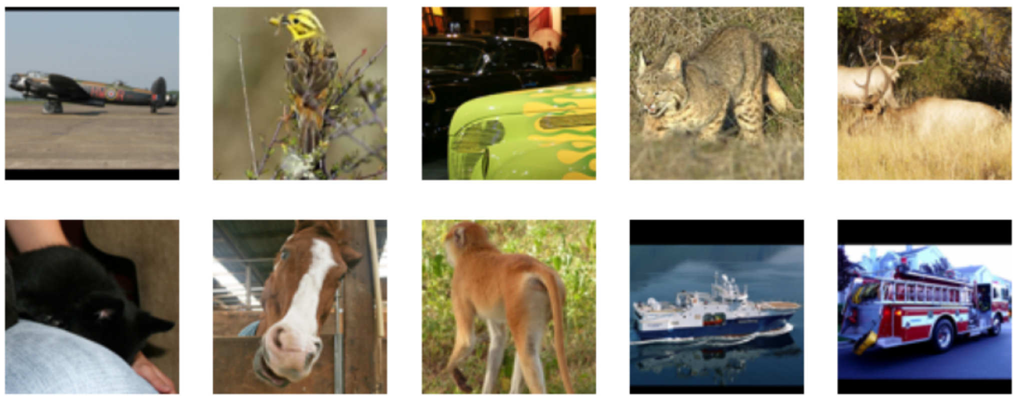


Figure 4.2: STL-10

deer, dog, frog, horse, ship, truck. There are 50000 training images and 10000 test images Figure 4.3.The dataset is divided into five training batches and one test batch, each with 10000 images.The test batch contains exactly 1000 randomly-selected images from each class.

Figure 4.3: CIFAR-10

Now we state the results obtained and analyse the performance of Image-Split on various neural network architectures-

4.2 Experiments and Results

Experiments in subsection 4.2.1,4.2.2 and 4.2.3 are done on the Intel Image Classification dataset.In subsection 4.2.4, the experiment is performed on all three datasets.

4.2.1 Single hidden layer

SInce Single hidden layered neural networks are relatively simple, we experiment with a wide range of splits.The images are split various sizes ranging from simple 2x2 to 10x10 splits.The number of single hidden layered networks used in the process can range from 4 to 100.Table 4.1 shows the results obtained for single layered networks. Figure 4.4 shows the scatter plot of the data in table 4.1.

The accuracy peaks around 0.80.This shows that more splits don't necessarily increase the accuracy.

Table 4.1: Performance of ImageSplit with single hidden layer networks

Networks	Accuracy
Single	0.39
2x2	0.80
3x3	0.81
5x5	0.83
6x5	0.82
6x6	0.83
10x5	0.82
10x6	0.78
15x5	0.79
10x9	0.78
10x10	0.81

4.2.2 Multiple hidden layers

Multi-layered networks with 3 hidden layers are used to process the split images.Table 4.2 shows the results obtained for multi-layered networks.

Table 4.2: Performance of ImageSplit with multiple hidden layer networks

Networks	Accuracy
Single	0.48
2x2	0.81
4x4	0.83
5x5	0.83

For the single network there is a huge jump in accuracy but in case of the split networks it is clear that more layers doesn't increase the accuracy.

4.2.3 Convolutions

Simple CNNs with 2-3 convolutional layers are used to process the split images.Table 4.3 shows the results obtained for convolutional neural networks. As compared to the previous experiment, there is an increase in accuracy for the single network.There is also a jump in accuracy for the split networks.

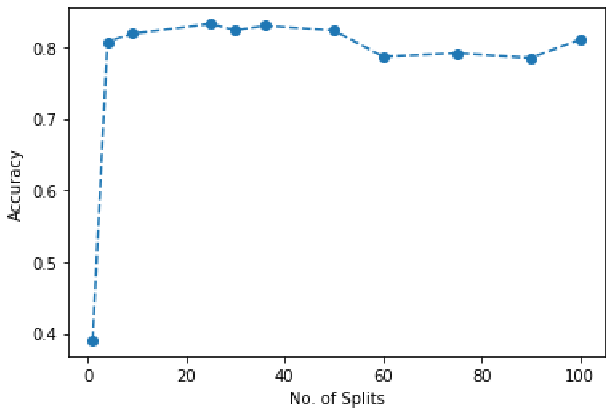


Figure 4.4: Single hidden layer network accuracy for different splits

Table 4.3: Performance of ImageSplit on CNN architectures

Networks	Accuracy
Single	0.75
2x2	0.87
4x4	0.87
5x5	0.85

4.2.4 Pre-trained models

Pre-trained models VGG16 and ResNet50 *(Keras API)* are used as feature extractors.Separately connected layers are added on top of the pre-trained feature extractors.These fully connected layers are the only layers which are trained from scratch in the training phase of the model.

VGG16

Table 4.4 shows the results obtained when using VGG16 and Table 4.5 for ResNet50.

Figure 4.5 shows the scatter plot of the data in table 4.4.

Table 4.4: Performance of ImageSplit with VGG16

Network Configuration	Accuracy		
	Intel Image	stl10	CIFAR-10
single	0.87	0.82	0.625
2x2	0.93	0.937	0.919
2x3	0.95	0.934	0.923
3x3	0.952	0.934	0.926

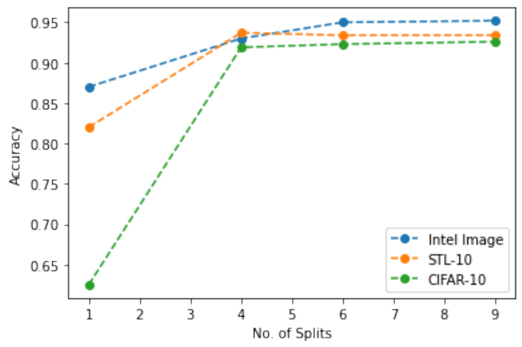


Figure 4.5: Performance of VGG16 for different splits on all three datasets

Performance of ResNet50 for different splits on all three datasets

Table 4.5 shows the results obtained when using ResNet50 with ImageSplit.Figure 4.6 shows the scatter plot of the data in table 4.5.

Table 4.5: Performance of ImageSplit with ResNet50

Network Configuration	**Accuracy**		
	Intel Image	**stl10**	**CIFAR-10**
single	0.88	0.85	0.393
2x2	0.94	0.951	0.933
2x3	0.95	0.934	0.930
3x3	0.96	0.940	0.925

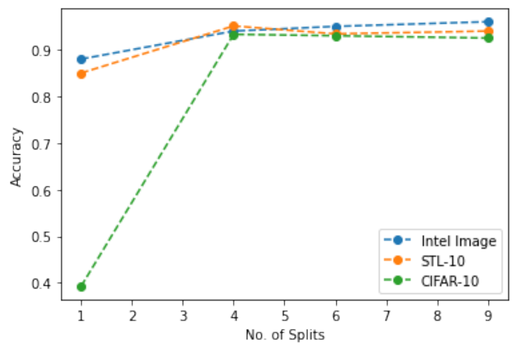


Figure 4.6: ResNet50 graph

There is an increase in accuracy for all three datasets for both VGG16 and ResNet50.The feature extractors, even though not trained exclusively for the given datasets, have shown a good accuracy.

4.3 Observations

In short, the weight multiplications in single matrices takes more computations as opposed to multiplication in multiple smaller matrices. The number of parameters reduce significantly when taking a divided input as compared to the whole input. For example,a CNN having three convo- lutional layers followed by two fully connected layers will have 15, 549, 254 parameters for a input image of the size 100x100. The same network has a size of 349, 254 parameters for a image size of 25×25. In total, for 4 smaller images, the total parameters used are 1, 397, 016. Thus the ImageSplit technique can reduce the computational complexity with an added advantage of improved performance in terms of detection accuracy.

Chapter 5

Conclusion

To summarize, this report explores solutions to the increasing computational complexity of deep learning architectures.There are existing complexity reduction techniques like pruning and splitting network architectures.The report delves deep into the working of splitting techniques like SplitNet, Split-CNN which are about splitting the convolutional layer in the CNN architecture.Further, it proposes *ImageSplit*, a novel technique which involves splitting images and processing them separately.This is method is different from existing splitting techniques as it involves splitting the image rather than the neural network architecture.

Processing multiple smaller parts of an image is computationally less intensive than processing the image as a whole.This insight is mathematically validated by comparing the number of operations in a matrix multiplication of a multiple smaller matrices versus a single big matrix.Splits of various sizes and shapes can be done, although too many splits need not necessarily increase the accuracy.Increasingly complex architectures give better results, that doesn't imply that they are desirable since the main goal is to reduce the overall number of computations performed.The results clearly show that this process has a much better accuracy for all the architectures, be it single layered networks or complex networks like VGG16 and ResNet50.Splitting and running the architectures in parallel might reduce the time taken to perform the classification task.

REFERENCES

[1] LeCun, Y., Bengio, Y. Hinton, G. Deep learning. Nature 521, 436–444 (2015). `https://doi.org/10.1038/nature14539`

[2] Alex Krizhevsky, Ilya Sutskever, and Georey E Hinton. 2012. Imagenet classication with deep convolutional neural networks. In Advances in Neural Information Processing Systems. 1097–1105

[3] Sutskever, Ilya, Vinyals, Oriol, and Le, Quoc V. Sequence to Sequence Learning with Neural Networks. In NIPS, pp. 3104–3112, 2014

[4] Alex Graves, Abdel-rahman Mohamed, and Georey Hinton. 2013. Speech recognition with deep recurrent neural networks. In 2013 IEEE International Conference on Acoustics, Speech and Signal Processing. IEEE, 6645–6649

[5] Hinton, Geoffrey, Deng, Li, Yu, Dong, Dahl, George, rahman Mohamed, Abdel, Jaitly, Navdeep, Senior, Andrew, Vanhoucke, Vincent, Nguyen, Patrick, Kingsbury, Brian, and Sainath, Tara. Deep Neural Networks for Acoustic Modeling in Speech Recognition. IEEE Signal Processing Magazine, 29:82–97, 2012.

[6] Ba, Jimmy and Caruana, Rich. Do Deep Nets Really Need to Be Deep? In NIPS, 2014.

[7] Alvarez, Jose M and Salzmann, Mathieu. Learning the Number of Neurons in Deep Networks. In NIPS. 2016.

[8] Dean, Jeffrey, Corrado, Greg S., Monga, Rajat, Chen, Kai, Devin, Matthieu, Le, Quoc V., Mao, Mark Z., Ranzato, MarcAurelio, Senior, Andrew, Tucker, Paul, Yang, Ke, and Ng, Andrew Y. Large Scale Distributed Deep Networks. In NIPS, 2012

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Je Dean. 2013.Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems. 3111–3119.

[10] Collins, Maxwell D. and Kohli, Pushmeet. Memory Bounded Deep Convolutional Networks, arXiv, 2014. `https://arxiv.org/abs/1412.1442`

[11] Han, Song Mao, Huizi Dally, William. (2016). Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.

[12] Alvarez, Jose M and Salzmann, Mathieu. Learning the Number of Neurons in Deep Networks. In NIPS, 2016

[13] Wen, Wei, Wu, Chunpeng, Wang, Yandan, Chen, Yiran, and Li, Hai. Learning Structured Sparsity in Deep Neural Networks. In NIPS. 2016.

[14] Dean, Jeffrey, Corrado, Greg S., Monga, Rajat, Chen, Kai, Devin, Matthieu, Le, Quoc V., Mao, Mark Z., Ranzato, MarcAurelio, Senior, Andrew, Tucker, Paul, Yang, Ke, and Ng, Andrew Y. Large Scale Distributed Deep Networks. In NIPS, 2012.

[15] Shankar, Sukrit, Robertson, Duncan, Ioannou, Yani, Criminisi, Antonio, and Cipolla, Roberto. Refining Architectures of Deep Convolutional Neural Networks. In CVPR, 2016

[16] Krizhevsky, Alex. One Weird Trick for Parallelizing Convolutional Neural Networks. arXiv:1404.5997, 2014.

[17] Chilimbi, Trishul, Suzue, Yutaka, Apacible, Johnson, and Kalyanaraman, Karthik. Project Adam: Building an Efficient and Scalable Deep Learning Training System. In OSDI, 2014.

[18] Zhang, Hao, Hu, Zhiting, Wei, Jinliang, Xie, Pengtao, Kim, Gunhee, Ho, Qirong, and Xing, Eric. Poseidon: A System Architecture for Efficient GPU-based Deep Learning on Multiple Machines. arXiv:1512.06216, 2015

[19] MalekHosseini, Emad and Hajabdollahi, Mohsen and Karimi, Nader and Samavi, Shadrokh and Shirani, Shahram. "Splitting Convolutional Neural Network Structures for Efficient Inference", arXiv, 2020, 10.48550/ARXIV.2002.03302

[20] Juyong Kim, Yookoon Park, Gunhee Kim, Sung Ju Hwang, "SplitNet: Learning to Semantically Split Deep Networks for Parameter Reduction and Model Parallelization", ICML, 2017

[21] Huth, Alexander H., Nishimoto, Shinji, Vu, An T., and Gallant, Jack L. A Continuous Semantic Space Describes the Representation of Thousands of Object and Action Categories across the Human Brain. Neuron, 76 (6):1210–1224, December 2012

[22] Jin, Tian and Hong, Seokin. "Split-CNN: Splitting Window-based Operations in Convolutional Neural Networks for Memory System Optimization", Association for Computing Machinery (2019), pp.835-847

[23] Sharma, Ankit Kumar and Foroosh, Hassan. "Slim-CNN: A Light-Weight CNN for Face Attribute Prediction",2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020), pp.329-335

[24] David Optiz, Richard Maclin, "Popular Ensemble Methods: An Empirical Study", Journal of Artificial Intelligence Research (1999), pp.169-198

[25] Leo Breiman, "Bagging Predictors", Technical Report No. 421 September 1994, Department of Statistics, University of California

[26] Tin Kam Ho, "Random Decision Forests", Proceedings of 3rd International Conference on Document Analysis and Recognition, 14-16 Aug. 1995

[27] Reprinted from blog post "Everything you need to know about VGG16', by Rohini G, 2021, Retrieved from `https://medium.com/@mygreatlearning/everything-you-need-to-know-about-vgg16-7315defb5918`.Copyright (2021) by Great Learning.

[28] Reprinted from blog post "ResNet50', by Aditi Rastogi, 2022, Retrieved from `https://blog.devgenius.io/resnet50-6b42934db431`.Copyright (2022) by DevGenius.

[29] Adam Coates, Honglak Lee, Andrew Y. Ng. "An Analysis of Single Layer Networks in Unsupervised Feature Learning" AISTATS, 2011. `http://cs.stanford.edu/ acoates/stl10`

[30] Intel Image Classification Dataset,
`https://www.kaggle.com/puneet6060/intel-image-classification`

[31] Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009. `http://www.cs.toronto.edu/ kriz/cifar.html`

[32] Stevo. Bozinovski and Ante Fulgosi (1976). "The influence of pattern similarity and transfer learning upon the training of a base perceptron B2." (original in Croatian) Proceedings of Symposium Informatica 3-121-5, Bled.

[33] Simonyan, Karen Zisserman, Andrew. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv 1409.1556.

[34] He, Kaiming Zhang, Xiangyu Ren, Shaoqing Sun, Jian. (2016). Deep Residual Learning for Image Recognition.770-778. 10.1109/CVPR.2016.90.