Certificate

This is to certify that the report entitled **"PRECLA.IO: An Image Prediction and Classification Engine"** has been undertaken and written under my supervision and it describes the original research work carried out by **Charul Rathore (140135), Aakanksha Kaushik (140144)** and **Gargee Sanyal (140209)** in Computer Science and Engineering for the degree of B. Tech. To the best of my knowledge and belief, this work is original and has not been submitted elsewhere for any degree from any other institution in India or abroad.

Dr. A. K. Singh
Assistant Professor, CSE

Prof. Anil Kumar
Professor and Head, CSE

Approval Certificate

This thesis report entitled **"PRECLA.IO: An Image Prediction and Classification Engine"** by **Charul Rathore (140135), Aakanksha Kaushik (140144)** and **Gargee Sanyal (140209)** is approved for the degree of B. Tech.

Examiner(s)

_____

_____

Supervisor(s)

_____

_____

Date: 08.05.2018

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea / data / fact / source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

We also attest that this report is free of plagiarism of any kind and that we are fully aware of the Plagiarism Prevention Policy of Mody University of Science and Technology.

Charul Rathore [140135]

Aakanksha Kaushik [140144]

Gargee Sanyal [140209]

Date: 08.05.2018

Acknowledgement

We are using this opportunity to express our gratitude to everyone who supported us throughout this project. We are thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. We are sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

We express our warm thanks to **Dr. A. K Singh**, our mentor, for his support and guidance at Mody University of Science and Technology. We would also like to thank all the people who provided us with the facilities being required and conductive conditions for our project.

Thank you.

Charul Rathore [140135]

Aakanksha Kaushik [140144]

Gargee Sanyal [140209]

Abstract

In all our day to day activities we classify things based on situations and on our needs. Human beings do classification of any kind by their natural perception. Classifying data is a common task in machine learning which requires artificial intelligence. Inception model and transfer learning are new technique suitable for generalized classification tasks. It has a set of learning methods used for classification, regression and outlier's detection. It is based on implemented layers of Convolutional Neural Networks.

Images can affect people on an emotional level as well. Since the emotions that arise in the viewer of an image are highly subjective, they are rarely indexed. However there are situations when it would be helpful if images could be retrieved based on their emotional content. We investigate and develop methods to extract and combine low-level features that represent the emotional content of an image, and use these for image emotion classification. Specifically, we exploit theoretical and empirical concepts from psychology and art theory to extract image features that are specific to the domain of artworks with emotional expression. For testing and training, we use three data sets: the International Affective Picture System (IAPS); a set of artistic photography from a photo sharing site (to investigate whether the conscious use of colors and textures displayed by the artists improves the classification); and a set of peer rated abstract paintings to investigate the influence of the features and ratings on pictures without contextual content. Improved classification results are obtained on the International Affective Picture System (IAPS), compared to state of the art work.

List of Contents

List of Figures

List of Tables

List of Graphs

Chapter 1

Introduction

Image classification is the task of taking an input image and outputting a class (a cat, dog, etc) or a probability of classes that best describes the image. For humans, this task of recognition is one of the first skills we learn from the moment we are born and is one that comes naturally and effortlessly as adults. Without even thinking twice, we're able to quickly and seamlessly identify the environment we are in as well as the objects that surround us. When we see an image or just when we look at the world around us, most of the time we are able to immediately characterize the scene and give each object a label, all without even consciously noticing. These skills of being able to quickly recognize patterns, generalize from prior knowledge, and adapt to different image environments are ones that we do not share with our fellow machines.

In recent years, one of the major applications of machine learning has been in image classification hence it is pointless to say how essential image classification/recognition is in the field of computer vision – image recognition is vital for bridging the huge semantic gap between an image, which is merely a scatter of pixels to untrained computers, and the object it presents. Therefore, there have been extensive research efforts on developing efficient image classifiers and visual object recognizers.

Classification helps us in taking our daily life decisions. Whenever an object is placed in a specific group or class depending upon the attributes corresponding to that object the classification is needed. Since numerous images are being produced every day we need to classify them for easier and faster accessibility as a result the majority of the industrial problems are classification problems. Scientists are trying to improve the classification accuracy by devising advanced classification techniques. The image categorization of images into various groups is the result of image processing done during the classification. For example, speech recognition, character recognition, stock market prediction, weather forecasting, bankruptcy prediction, medical diagnosis, etc. in these areas, classification problems can be solved mathematically in a non linear fashion. Because of the accuracy and

distribution of data properties and model capabilities, this type of problem solving is relatively tricky. For efficient and rapid analysis of the surroundings, the categorization of the scene is required. It is difficult to classify a scene if it contains blurry and noisy content. It is quite challenging to identify an object in an image if the images are affected due to noise, poor quality, occlusion or background clutter and whenever an image consists of multiple objects the challenge gets multiplied. There has been a steady rise in new classification algorithms, techniques in recent years. Hence, we need an algorithm for faster and easier image classification.

1.1 Scope

In recent years, there is a growing consensus that it is necessary to build general purpose object recognizers that are able to recognize many different classes of objects. Image object classification and detection are two of the most essential problems in computer vision. They are the basis of many other complex vision problems, such as segmentation, tracking, and action analysis. Object classification and detection is computer vision, pattern recognition and very active research direction in the field of machine learning. Object classification and detection is widely used in many fields, including face recognition, and pedestrian detection in the field of security, intelligent video analysis, the pedestrian tracking, object recognition, vehicle traffic scene in the field of traffic count, retrograde motion detection, license plate detection and recognition, and the Internet in the field of content-based image retrieval, photo album automatic classification and so on. Can say, object classification and detection has been applied to every aspect of People's Daily life, computer automatic classification and detection technology also to a certain extent reduce the burden of people, changed the human way of life. Image classification is a key task in many areas, so how to use the computer quickly and accurately classify the image identifying technology got many scholars pay attention to. Hence image classification has become a crucial and challenging task in various application domains, including remote sensing, vehicle navigation, biomedical imaging, video-surveillance, biometry, industrial visual inspection, robot navigation, and vehicle navigation.

Image processing is being applied in many fields in today's world:

- Automotive sector: In developing advanced drivers assist for semi-autonomous cars and also heavily used in autonomous/driver-less cars.

- Image enhancing: The camera apps in smart phones and digital cameras using image processing to enhance the image quality, video stabilization and noise removal etc.
- Robotics: Mobile robot's navigation in unknown environment (SLAM), control of the robot by processing the video feed from the camera on robot to extract the live scene around it.
- Gaming: Advanced gaming consoles like Xbox kinect uses image processing from motion analysis of the human player.
- Problem specific solutions: image processing is used as a solution to a variety of problems, starting from facial recognition access to defects identification in manufacturing industries.
- Manufacturing: To identify defects in the processes and also to control the robots in performing certain tasks. For eg defects in manufacturing of a Printed Circuit Board (PCB) can be observed using high resolution image processing.
- Human machine interface: machines are made smart by adding gestural interface, or human action response interfaces, which decodes the actions of the human user to perform certain tasks.

Camera simulates the eyes of a human being, which is one of the main sensor of the human body using which brain takes decisions. Camera is relatively cheap and decoding the image from it can give enormous amount of information which can be used to perform certain actions/tasks. So Image processing is one of the emerging and is a future tool, so it has lot of scope.

1.2 Technologies Used

There were a lot of attempts in 80's and 90's to classify the images and all of them tried a similar approach which was to think about the feature that make up an image and hand code detectors for each of them. But there is so much variety out there for example no two apples look exactly the same so the results were always terrible. This was considered a task only we humans can do. But in 1998 a researcher introduced a model called 'Convolutional Neural Networks (CNNs)', which was capable of classifying characters with 99% accuracy. CNNs is a special type of Neural Networks that works in the same way of a regular neural network except that it has a convolution layer at the beginning. CNNs learned features by itself. In 2012 it was used by another researcher named Alex Krizhevsky at the yearly ImageNet

competition which is basically the annual Olympics of computer vision and it was able to classify thousands of images with a new record accuracy of 85%. Since then CNNs have been adopted by Google to identify photos in search, Facebook for automatic tagging, Amazon for their product recommendations, Pinterest for their home feed personalization, and Instagram for their search infrastructure.

In this project we'll be using Python and TensorFlow to write the program. Python is an interpreted high-level programming language created by Guido van Rossum for general-purpose programming. It contains special libraries for machine learning namely scipy and numpy, which is great for linear algebra and getting to know kernel methods of machine learning. TensorFlow is an open source deep learning framework created by Google that gives developers granular control over each neuron (known as a "node" in TensorFlow) so the developer can adjust the weights and achieve optimal performance. TensorFlow has many built-in libraries (few of which we'll be using for image classification).

We will be using a pre-trained CNN model called Inception. Inception was trained by Google on 100k images with a thousand categories to provide state of the art performance on the ImageNet Large-Scale Visual Recognition Challenge and to be more computationally efficient than its competitor architectures. However, what makes Inception exciting is that its architecture can be applied to a whole host of other learning problems in computer vision. . The Inception model performs a series of operations on that data until it outputs a label and classification percentage. Each layer is a different set of abstractions. In the first layer it basically taught itself Edge Detection then Shape Detection in the middle layers and they get increasingly more abstract up until the end. We will be using the concept of Transfer learning. It makes use of the knowledge gained while solving one problem and applying it to a different but related problem i.e. applying the learning from a previous training session to a new training session.

Finally, these concepts are encapsulated into one model which is our classifier. A casual flower dataset is used to train and test the model in initial phases which gave us a brief idea of how features were picked in our model. Later, we exploit theoretical and empirical concepts from psychology and art theory to extract image features that are specific to the domain of art works with emotional expressions. We investigate and develop methods to extract and combine low-level features that represent the emotional content of an image, and use these for image emotion classification.

Chapter 2

Review of Literature

Image classification, being considered as one of the most complex areas in image processing, has several methods to classify images and provide appreciable classification results. But the presence of blurry and noisy content makes it complex and difficult to classify and that is where these available methods fail to provide competent classification. Supervised and unsupervised classifications are the two major image classification methods and each classification has its own advantages and disadvantages.

The main objective of this literature survey is to provide a perfunctory overview about some of most prominent image classification methods and a brief comparison between them. The main objective of image classification is to identify the features occurring in an image either via supervised or unsupervised methods. The type of methods to be used for classification depends on the available datasets. Trained database and human annotation is needed in supervised classification while in unsupervised classification, human annotation is not a must since it is mostly automated by the computer [10].

2.1 Literature Survey

Noridayu et al. [1], proposed a new approach for improving performance of object class recognition by combining different features with local features. In this, the features are extracted from the image, which are boundary-based shape features and local features. The first type of feature is based on the outline of segmented objects while the second are based on the interior information of objects. Two features thus obtained are combined and then concatenating those features in a new single feature vector by using feature fusion approach. Then features are classified by using Support Vector Machine. The classification accuracy is 70%.

Yasuo et al. [2], increased the performance of global features by using local feature correlation and then classifying scene. First local features are extracting from the image. It involves two steps, key point detection based on grid and feature description using SIFT

descriptor. Next classification of scene is based on Linear Discriminant Analysis (LDA). The average classification accuracy rate is 70%. This method does not classify objects more clearly, they only classify scenes. The disadvantage is that method is poor in object recognition.

Shanmugam et al. [4], classifying war scene from the natural scene by extracting wavelet features. By using after extracting wavelet features they are classified by using Artificial Neural Network and then Support Vector Machines (SVM). This paper also compares Artificial Neural Network and Support Vector machine and determining which one is best to classify war scene. First from the input image wavelet features are extracted and then that extracted features are given to normalization in order to maintain the data so that performance of classifier can be improved. Normalized features are given as input to Artificial Neural Network and also to Support Vector Machine. ANN classify the image using backward propagation algorithm and SVM classify the image using radial basis kernel function with p=5. In the case of SVM, it gives only 59% classification rate and in the case of ANN, it gives only 72.5% classification rate. Thus ANN provides good classification result in classifying war scene by extracting wavelet features when compared with SVM.

Vogel and B.Schiele [5], proposed a novel image representation to access natural scenes by local semantic description. They use a spatial grid layout which split the images into regular sub regions. The techniques use both colour and texture to perform landscape image classification and retrieval based on a two stage system. First the image is partitioned into 10×10 sub regions and each one is classified using K-NN or SVM. An image is then represented by a so-called Concept Occurrence Vector (COV) which measures the frequency of different objects in a particular image. Given the image representation a prototypical representation for each scene category can be learnt. Image classification carried out by using the prototypical representation itself or Multi SVM approach. The advantage of this approach is that they it uses human meaning to classify the object and then image and able to classify image into big number of categories. The average classification accuracy is 71.1%. The disadvantage of this approach is that here image classification is based on object occurrence so a wrong object classification will result into erroneous image classification.

Ponce et al. [9], proposed a spatial pyramid matching for recognizing natural scene categories. This technique works by repeatedly subdividing the image and computing histograms of local features at increasingly fine resolution and taking a weighted sum of number of matches that

occur at each level of resolution (L). This simple operation significantly improves performance over a basic bag-of-features representation. Two features extracted first one is "weak features," which are oriented edge points, i.e., points whose gradient magnitude in a given direction exceeds a minimum threshold. Second one is higher dimensional "strong features," which are SIFT descriptors of 16×16 pixel patches computed over a grid with spacing of 8 pixels. The pyramid matching works by placing a sequence of increasingly coarser grids over the feature space and taking a weighted sum of the number of matches that occur at each level of L. The two points are said to match at any fixed resolution if they belongs to same cell of the grid. The resulting "spatial pyramid" is a simple and computationally efficient extension of an order less bag-of-features image representation, and it reduces to standard bog- of-words when L=0. Multi-class classifier is done with SVM. This method achieves high accuracy on a large database of 15 natural scene categories and the well known Caltech-101 dataset. The classification rate is 72.2%.

The advantages of supervised classification are operator can detect errors and often remedy them. The disadvantage of this approach are training data can be time consuming and costly and also training data selected by the analyst, may not be representative of conditions encountered throughout the image. Another disadvantage of supervised classification is it is prone to human error.

Selim et al. [5], proposed a new method for classifying outdoor scenes. First, by using one-class classification and patch-based clustering algorithms images are partitioned into regions. Secondly, to obtain region types codebook, the resulting regions thus obtained are clustered, and then two models are constructed for scene representation: a bag of individual regions representation where each region is considered separately, and a bag of region pairs representation this means regions with same spatial relationships are considered together. Given these representations, scene classification was done using Bayesian classifiers. The advantage of this approach is that the proposed models significantly outperform global feature-based techniques. The correct classification rate is 62%.

Fei-Fei et al. [6], proposed a new approach to classify events into static images by integrating scene and object categorization. The technique used to categorize scene and object is an integrative model. By using this model they are extracting local features from the image and then categories object and after categorizing object they recognize the scene and then by integrating both scene and object recognition events are classified. For example if an event is

given such as rowing, in order to identify this event as rowing firstly identify the objects such as tree, water, athlete and then label scene as lake and finally by integrating both scene and object classification, that particular event is identified as rowing. Thus this model is trying to tell three answers: what, where and who. The model can classify the events correctly at 73.4% accurately. The disadvantage of this paper is that by scene or object categorization alone cannot achieve this performance. Fei-Fei and P. Perona [7], proposed a method to classify natural scene. The images of scene are represented by a collection of local regions. These local regions are denoted as code words. Then these code words are automatically distributed to each local patch. Then identify a model that represents the distribution of these code words in each category of scenes. In recognition, they first identify all the code words in the unknown image. Then find in which category model distribution of these code words of that particular image belongs to. Thus they classify images. In this classification scenes are categorized in the training phase itself so time will be less when recognizing unknown images and also here code words are automatically distributed to each local patches that extracted from the image. The images are correctly classified at 76%.

Jiang et al. [8], proposed a scene oriented hierarchical classification of blurry and noisy images. Three strategic approaches used are global pathway for essential capture, local pathway for highlight detection and thirdly hierarchical classification. In this firstly extracting global features by using Gabor filter, from that Gabor image extract only real part and then applying Principle Component Analysis (PCA) to reduce dimensionality. Then get the visual context by combining real part of Gabor image and PCA. Secondly, pseudo restoration is done directly on blurry and noisy images and from that pseudo restored image highlight detection ii set of local features has been found and then extract conspicuous local features by using Harris Affine detector and thirdly combining both features by using log linear model and clustered by using Monte Carlo approach. Finally, these clustered features are classified by using Self Organizing Tree Algorithm (SOTA). The classification rate is 81.95%.

The advantage of unsupervised method are time taken is less and minimize human errors and also no extensive or detailed a priori knowledge of the region is required. The disadvantage of this method are maximally-separable clusters in spectral space may not match our perception of the important classes on the landscape and also limited control over the “menu” of classes.

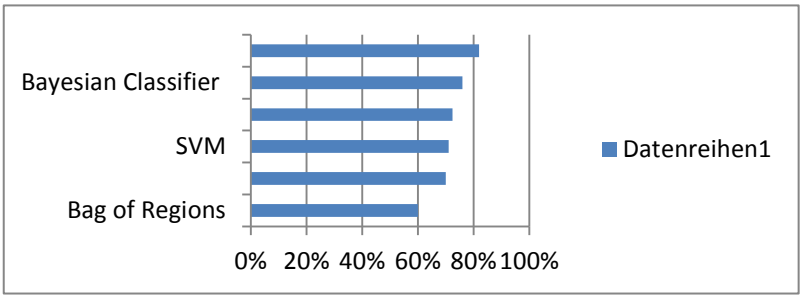
The classification rate of various image classification methods are shown in the table 1. It is shown that Self Organizing Tree Algorithm classify images at the rate of 81.95% even it

contain blurry and noisy content when compared with other image classification methods. In the graph 1 comparison of classification rate for different classification methods are also shown. In the X-axis classification methods and in the y-axis classification rate is shown.

Table 2.1 Classification Rate of various Image Classification Methods

METHODS	CLASSIFICATION RATE
Bag of Regions	60%
Linear Discriminant Analysis (LDA)	70%
Support Vector Machine (SVM)	71.1%
Artificial Neural Network (ANN)	72.5%
Bayesian Classifier	76%
Self-Organising Tree Algorithm	81.95%

Graph 2.1 Comparison of Classification Rate for different Classification Methods



2.2 Research Gap

Object recognition systems constitute a deeply entrenched and omnipresent component of modern intelligent systems. Research on object recognition algorithms has led to advances in factory and office automation through the creation of optical character recognition systems, assembly-line industrial inspection systems, as well as chip defect identification systems. It has also led to significant advances in medical imaging, defence and biometrics. The breadth of approaches adopted over the years in attempting to solve the problem, and highlight the important role that active and attentive approaches must play in any solution that bridges the semantic gap in the proposed object representations, while simultaneously leading to efficient learning and inference algorithms. From the earliest systems which dealt with the character recognition problem, to modern visually-guided agents that can purposively search entire

rooms for objects, we argue that a common thread of all such systems is their fragility and their inability to generalize as well as the human visual system can [11].

In 1965, Gordon Moore stated that the number of transistors that could be incorporated per integrated circuit would increase exponentially with time. This provided one of the earliest technology roadmaps for semi-conductors. Even earlier, Engelbart made a similar prediction on the miniaturisation of circuitry. Engelbart would later join SRI and found the Augmentation Research Center (ARC) which is widely credited as a pioneer in the creation of the modern Internet era computing, due to the center's early proposals for the mouse, video conferencing, interactive text editing, hypertext and networking. As Engelbart would later point out, it was his early prediction on the rapid increase of computational power that convinced him on the promise of the research topics later pursued by his ARC laboratory. The early identification of trends and shifts in technology can provide a competitive edge for any individual or corporation. The question arises as to whether we are currently entering a technological shift of the same scope and importance as the one identified by Moore and Engelbart fifty years ago.

For all intents and purposes, Moore's law is coming to an end. While Moore's law is still technically valid, since multicore technologies have enabled circuit designers to inexpensively pack more transistors on a single chip, this no longer leads to commensurate increases in application performance. Moore's law has historically provided a vital technology roadmap that influenced the agendas of diverse groups in academia and business. Today, fifty years after the early research on object recognition systems, we are simultaneously confronted with the end to Moore's law and with a gargantuan explosion in multimedia data growth. Fundamental limits on processing speed, power consumption, reliability and programmability are placing severe constraints on the evolution of the computing technologies that have driven economic growth since the 1950s. It is becoming clear that traditional von-Neumann architectures are becoming unsuitable for human-level intelligence tasks, such as vision, since the machine complexity in terms of the number of gates and their power requirements tends to grow exponentially with the size of the input and the environment complexity. The question for the near future is that of determining to what extent the end to Moore's law will lead to a significant evolution in vision research that will be capable of accommodating the shifting needs of industry. As the wider vision community slowly begins to address this fact, it will define the evolution of object recognition research, it will influence the vision systems that

remain relevant, and it will lead to significant changes in vision and computer science education in general by affecting other related research areas that are strongly dependent on vision (such as robotics).According to the experts responsible for the International Technology Roadmap for Semiconductors, the most promising future strategy for chip and system design is that of complementing current information technology with low-power computing systems inspired by the architecture of the brain. How would von-Neumann architectures compare to a non von-Neumann architecture that emulates the organization of the organic brain? The two architectures should be suitable for complementary applications. The complexity of neuromorphic architectures should increase more gradually with increasing environment complexity, and it should tolerate noise and errors. However, such neuromorphic architectures would likely not be suitable for high precision numerical analysis tasks. Modern von-Neumann computing precipitates the need for a program that relies on synchronous, serial, centralized, hardwired, general purpose and brittle circuits. The brain architecture on the other hand relies on neurons and synapses operating in a mixed digital-analog mode, is asynchronous, parallel, fault tolerant, distributed, slow, and with a blurred distinction between CPU and memory (as compared to von-Neumann architectures) since the memory is, to a large extent, represented by the synaptic weights. How does our current understanding of the human brain differentiate it from typical von-Neumann architectures? Turing made the argument that since brains are computers then brains are computable. But if that is indeed the case, why do reliable image understanding algorithms still elude us? Churchland and Hawkins argue that general purpose AI is difficult because

- Computers must have a large knowledge base which is difficult to construct, and because
- It is difficult to extract the most relevant and contextual information from such a knowledge base.

As it was demonstrated throughout our discussion on object recognition systems, the problem of efficient object representations and efficient feature extraction constitutes a central tenet of any non-trivial recognition system, which supports the viewpoint of Churchland and Hawkins.There is currently a significant research thrust towards the construction of neuromorphic systems, both at the hardware and the software level. This is evidenced by recent high-profile projects, such as EU funding of the human brain project with over a billion

Euros over 10 years, U.S. funding for the NIH BRAIN Initiative, and by the growing interest in academia and industry for related projects. The appeal of neuromorphic architectures lies in

- The possibility of such architectures achieving human like intelligence by utilizing unreliable devices that are similar to those found in neuronal tissue,
- The ability of neuromorphic strategies to deal with anomalies, caused by noise and hardware faults for example, and
- Their low-power requirements, due to their lack of a power intensive bus and due to the blurring of a distinction between CPU and memory.

Vision and object recognition should assume a central role in any such research endeavour. About 40% of the neocortex is devoted to visual areas V1, V2, which in turn are devoted just to low-level feature extraction. It is thus reasonable to argue that solving the general AI problem is similar in scope to solving the image understanding problem (see Sec.1). Current hardware and software architectures for vision systems are unable to scale to the massive computational resources required for this task. The elegance of the solution to the vision problem is astounding. The human neocortex consists of 80% of the human brain, which has around 100 billion neurons and 1014 synapses, consumes just 20-30 Watts, and is to a large extent self trained. One of the most astounding results in neuroscience is attributable to Mountcastle. By investigating the detailed anatomy of the neocortex, he was able to show that the micro-architecture of the regions looks extremely similar regardless of whether a region is for vision, hearing or language. Mountcastle proposed that all parts of the neocortex operate based on a common principle, with the cortical column being the unit of computation. What distinguishes different regions is simply their input (whether their input is vision based, auditory based etc.).

From a machine learning perspective this is a surprising and puzzling result, since the no-free-lunch theorem, according to which it is best to use a problem specific optimization/learning algorithm, permeates much of the machine learning research. In contrast the neocortex seems to rely on a single learning architecture for all its tasks and input modalities. Looking back at the object recognition algorithms surveyed in this paper, it becomes clear that no mainstream vision system comes close to achieving the generalization abilities of the neocortex. This sets the stage for what may well become one of the most challenging and rewarding scientific endeavours of this century.

2.3 Fulfillment

Compared to other region proposal classification networks (fast RCNN) which perform detection on various region proposals and thus end up performing prediction multiple times for various regions in a image, Yolo architecture is more like FCNN (fully convolutional neural network) and passes the image (n×n) once through the FCNN and output is (m×m) prediction. This architecture is splitting the input image in mxm grid and for each grid generation 2 bounding boxes and class probabilities for those bounding boxes. Note that bounding box is more likely to be larger than the grid itself. Object detection is reframed as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities [12].

A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimizes detection performance. This unified model has several benefits over traditional methods of object detection. First, YOLO is extremely fast. Since frame detection is a regression problem, there's no need of a complex pipeline. Neural network is simply run on a new image at test time to predict detections. Base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means it can process streaming video in real-time with less than 25 milliseconds of latency.

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it implicitly encodes contextual information about classes as well as their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

Third, YOLO learns generalizable representations of objects. When trained on natural images and tested on artwork, YOLO outperforms top detection methods like DPM and R-CNN by a wide margin. Since YOLO is highly generalizable it is less likely to break down when applied to new domains or unexpected inputs.

The network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes across all classes for an image simultaneously. This means the network

reasons globally about the full image and all the objects in the image. The YOLO design enables end-to-end training and realtime speeds while maintaining high average precision.

System divides the input image into an S × S grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally we define confidence as Pr(Object) ×IOU . If no object exists in that cell, the confidence scores should be zero. Otherwise we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally the confidence prediction represents the IOU between the predicted box and any ground truth box. Each grid cell also predicts C conditional class probabilities, Pr(Classi|Object). These probabilities are conditioned on the grid cell containing an object. We only predict one set of class probabilities per grid cell, regardless of the number of boxes B. At test time we multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Classi}|\text{Object}) \times \Pr(\text{Object}) \times \text{IOU} = \Pr(\text{Classi}) \times \text{IOU}$$

which gives us class-specific confidence scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

Network Architecture and Training:

Changes to loss functions for better results is interesting. Two things stand out:

- Differential weight for confidence predictions from boxes that contain object and boxes that don’t contain object during training.
- Predict the square root of the bounding box width and height to penalize error in small object and large object differently.

The network has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, we simply use 1×1 reduction layers followed by 3×3 convolutional layers

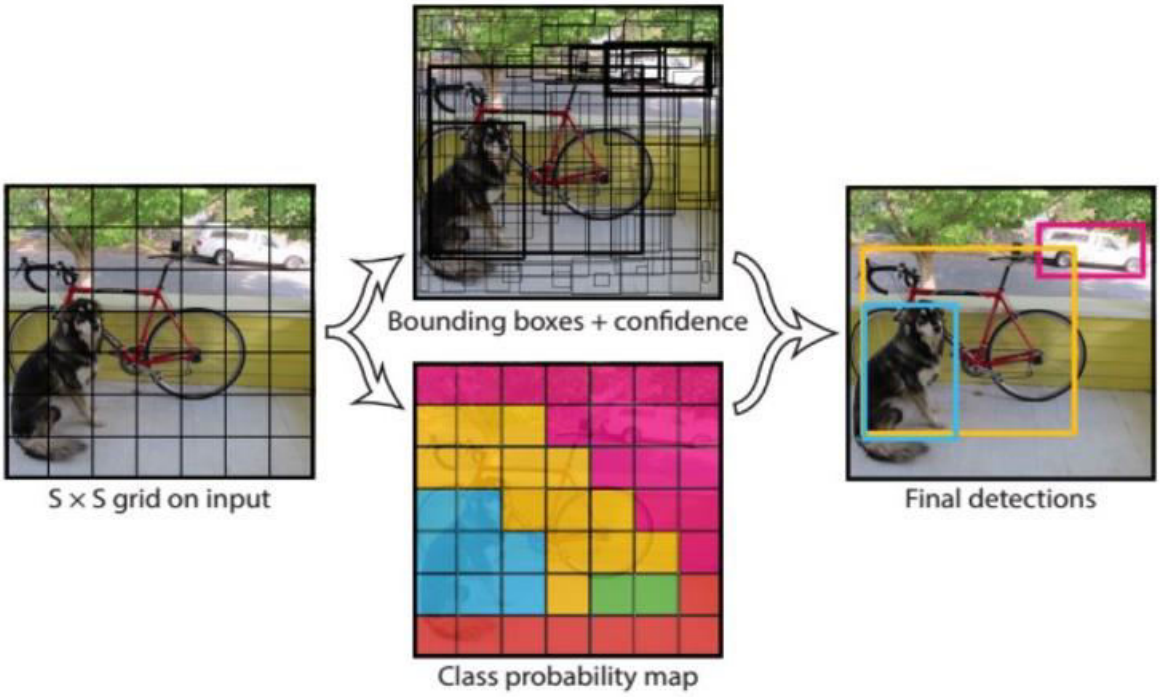


Fig 2.1: The Model. Our system models detection as a regression problem. It divides the image into an S×S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S×S× $(\boldsymbol{B \times 5 + C})$ tensor.

Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

Optimization for sum-squared error in the output of our model is done. Sum-squared error is sued because it is easy to optimize, however it does not perfectly align with the goal of maximizing average precision. It weights localization error equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the "confidence" scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on. To remedy this, increment in the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects are done. Two parameters are used, λcoord and λnoobj to accomplish this. We set λcoord = 5 and λnoobj = .5.

Sum-squared error also equally weights errors in large boxes and small boxes. Error metric should reflect that small deviations in large boxes matter less than in small boxes. To partially

address this, predicting the square root of the bounding box width and height is done instead of the width and height directly.

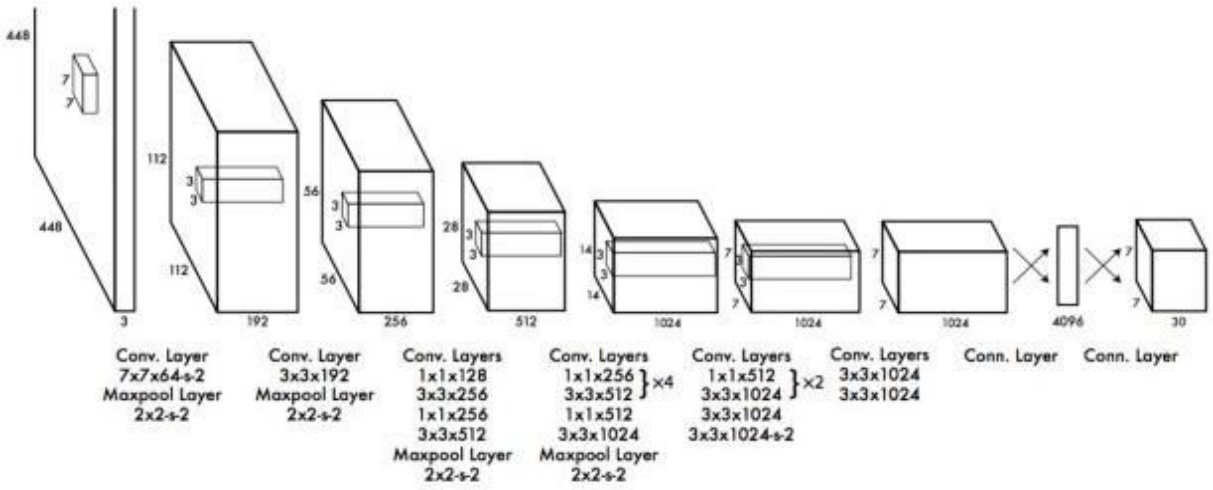


Fig 2.2: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preseding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

YOLO predicts multiple bounding boxes per grid cell. At training time we only want one bounding box predictor to be responsible for each object. Thus one predictor is assigned to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

Chapter 3

Design and Implementation

This module discusses the tools and frameworks used for developing the classifier, their installation process in brief (for Windows OS), libraries used. Since our aim was to design the classifier in Python, hence, it was the most important pre-requisite on our machine. In this chapter, the first few sections discuss installation of appropriate tools in brief followed by designing scheme and algorithm pattern of our classifier. Next few sections have descriptive explanation of the algorithm used along with background architecture and important functions. Finally, the chapter concludes with procedure on creating and training datasets.

3.1 Setting the Environment

The first step is to install Docker Toolbox. It is supported by Windows 7 and higher.

3.1.1 Docker Toolbox

To run Docker, the machine must have a 64-bit operating system running Windows 7 or higher. Additionally, virtualization should be enabled on the machine. One of the advantages of the newer Docker for Windows solution is that it uses native virtualization and does not require VirtualBox to run Docker. It provides a way to use Docker on Windows systems that do not meet minimal system requirements for the mobilenet applications. It includes the following tools:

- Docker CLI client for running Docker Engine to create images and containers
- Docker Machine so you can run Docker Engine commands from Windows terminals
- Docker Compose for running the `docker-compose` command
- Kitematic, the Docker GUI
- the Docker QuickStart shell preconfigured for a Docker command-line environment
- Oracle VM VirtualBox

Because the Docker Engine daemon uses Linux-specific kernel features, it cannot run natively on Windows. Instead, the Docker Machine command must be used, `docker-machine`, to create and attach to a small Linux VM on the machine. This VM hosts Docker Engine on Windows system. The following are the steps to install Docker but before installation, if the VirtualBox is running, it must be shut down first. Then go to Docker Toolbox page and download the installer. When the installer is launched, it opens "Setup – Docker Toolbox" dialog. If Windows security dialog prompts to allow the program to make a change, choose yes. The system displays the Setup - Docker Toolbox for Windows wizard. Press Next to accept all the defaults and then install. Accept all the installer defaults. The installer takes a few minutes to install all the components. When notified by Windows Security, the installer will make changes; make sure to allow the installer to make the necessary changes. The installer adds Docker Toolbox, VirtualBox, and Kitematic to your Applications folder.

Click the Docker QuickStart icon to launch a pre-configured Docker Toolbox terminal [22]. If the system displays a User Account Control prompt to allow VirtualBox to make changes to the computer; choose yes. The terminal does several things to set up Docker Toolbox and when it is done, the terminal displays the **$** prompt. The terminal runs a special 'bash' environment instead of the standard Windows command prompt. The 'bash' environment is required by Docker. Make the terminal active by clicking your mouse next to the $ prompt. Finally, the Docker is successfully installed and the terminal window looks like Fig 3.1.

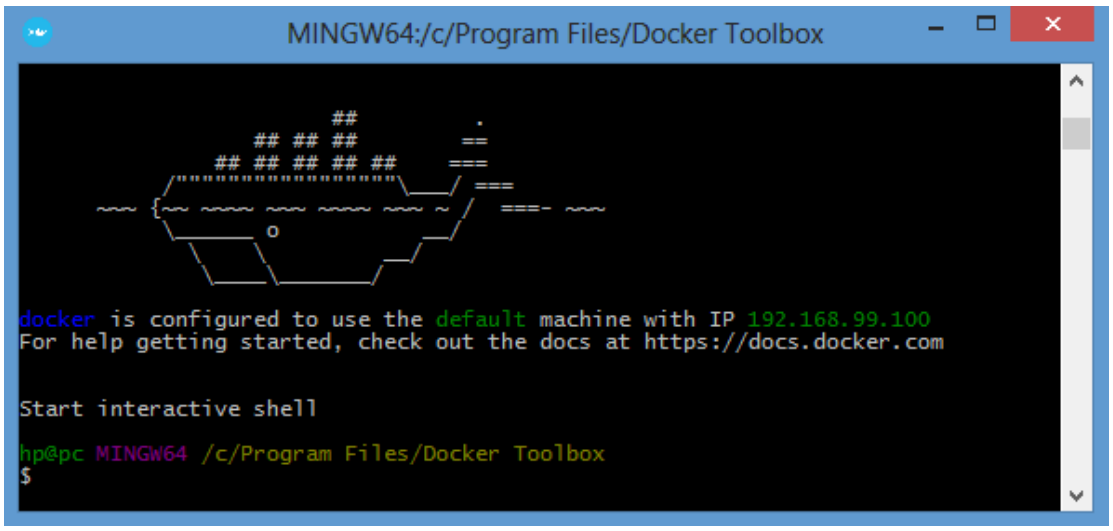


Fig 3.1 Docker terminal window for 'bash' environment

Now open the Docker Quickstart Terminal. Docker Toolbox creates a default Docker machine. The Docker machines available by typing can be viewed by using the command `docker-machine ls` which will display

```
NAME ACTIVE    DRIVER    STATE     URL
SWARM     DOCKER    ERRORS
default   *   virtualbox   Running   tcp://192.168.99.100:2376
v1.12.3
```

Now create a second (new) Docker machine named ‘vdocker’ by typing:

```
$ docker-machine create vdocker -d virtualbox
```

Type `docker-machine ls` at Docker Toolbox command Prompt to see the two Docker machines, the default one and the vdocker that was created in last steps:

```
$ docker-machine ls
NAME      ACTIVE   DRIVER       STATE     URL
SWARM   DOCKER    ERRORS
default   *        virtualbox   Running
tcp://192.168.99.100:2376
v1.12.3
vdocker   -        virtualbox   Running
tcp://192.168.99.101:2376
v1.12.3
```

Docker machine called ‘vdocker’ has been created and configured. The next step is to install TensorFlow which is discussed in the following section.

3.1.2 Tensorflow Installation

For this, open a Windows command prompt and enter the following command:

```
FOR /f "tokens=*" %i IN ('docker-machine env --shell cmd
vdocker') DO %i
```

The Docker machine will output similar to screenshot shown in Fig 3.2. Since Docker usually needs to have the same operating system on the Docker host system as it’s in the Docker container, a Linux VM to run a Docker container that is based on Linux under Windows is needed. Docker itself opens port 8888 between the VM and the container. It is crucial to explicitly forward the port from the outside of the VM to the open Docker container port by using said parameter –p 8888:8888.

```
docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow
```

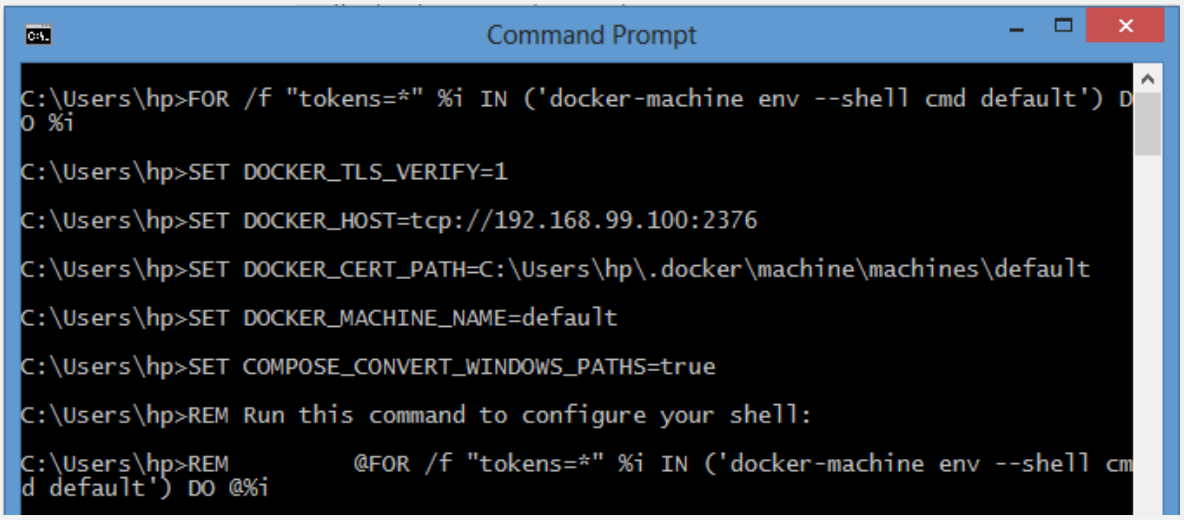


Fig 3.2 Docker machine output after windows command fired on its prompt

And it should start downloading the image which looks like Fig 3.3.

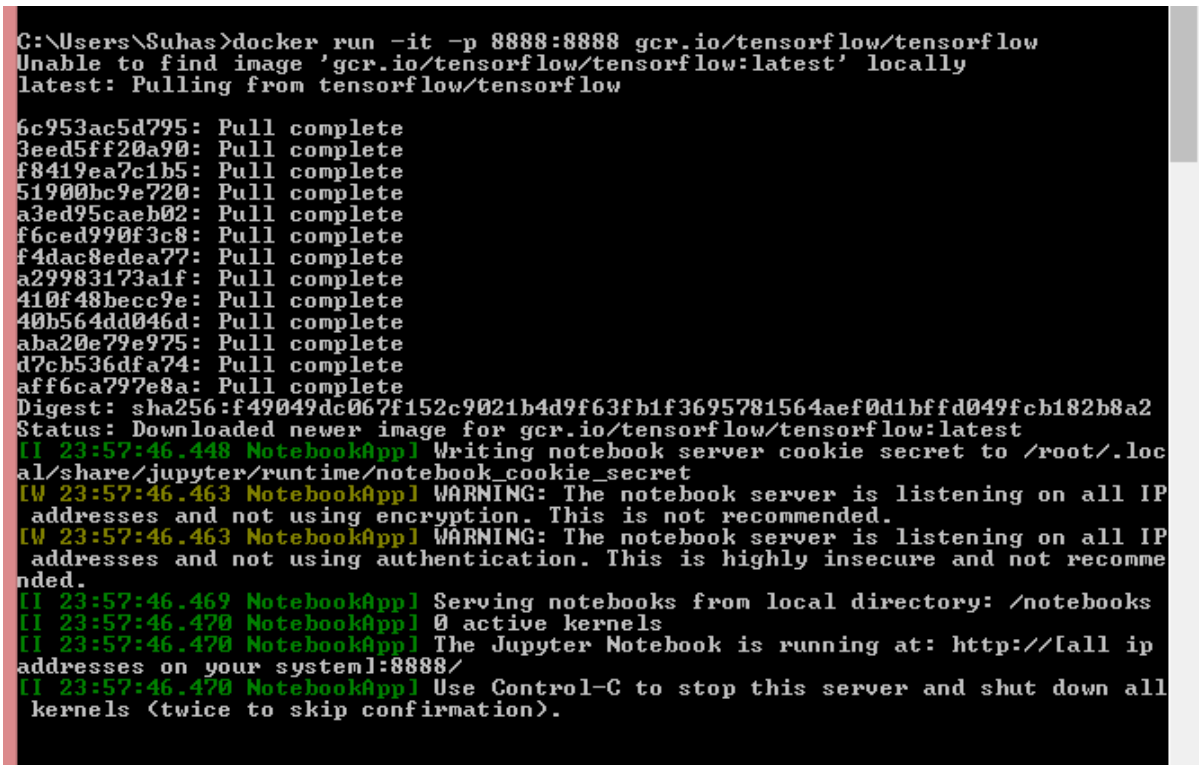


Fig 3.3 Launching Tensorflow from Windows Command Line

The option `-p 8888:8888` is used to publish the Docker container's internal port to the host machine, in this case to ensure Jupyter notebook connection. The format of the port mapping is `hostPort:containerPort`. Any valid port number for the host port can be specified but should use 8888 for the container port portion. Once Docker has pulled all the images from `gcr.io/tensorflow/tensorflow` and uncompressed, a Linux shell is obtained.

Now finally, launch the Jupyter engine using the following command, which launches the Jupyter engine on http://192.168.99.101:8888.

```
docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow
C:\Users\Suhas\Anaconda2\Scripts\jupyter.exe
```

3.2 Architectural Design

A high level approach to image classification uses a "Feature" to differentiate and classify them. A feature is an intelligent characteristic about the objects which help the computer understand and make a better decision to classify them. For example, if a water bottle is to be classified, machine should look for features like shape, contrast, height, width, transparency or even the label of the bottle [23]. Similarly, if machine needs to recognize faces, features like skin, hair or eye color etc are considered as very important characteristics. Obviously, these characteristics or features differ depending on the object that needs to be identified but none the less, each of them individually forms a single feature. These features are handed to a computer which trains itself by looking at them and then apply them to a training set of images. A training set is a set of more than a hundred, thousands or millions of new images, all of which contains the object that needs to be identified [25]. Over time, by training the machine with training set for a long stretch, it slowly starts picking up minute difference in the features that make one object differ from the other. The result of such training is a model that can probabilistically determine whether or not an image contains the object that needs to be identified. Crafting these features and building image detection algorithm not only take a lot of time and computational power but it also requires a lot of dense calculus. So to develop our classifier, we used a process called transfer learning in which we take pre-built model which has already been train of a set of images and then apply it to our case. The model has been trained by the Google's ImageNet database which contains information based on 1000 different classes like Dalmatians, Golden Retriever, Coffee mug, Laundry Machine etc, which acted as base for our classifier [32]. But before creating the model using transfer learning, we perform a thorough study on simple linear model and inception model. The following sections discuss these models in brief.

3.2.1 Simple Linear Model

A simple mathematical model is defined in tensorflow and an MNIST dataset of images is loaded in it.

The MNIST data-set consists of 70,000 images and associated labels (i.e. classifications of the images). The data-set is split into 3 mutually exclusive sub-sets: Training set, Test set and Validation set. Only first two sets are used for this project.

The data-set has been loaded as One-Hot encoding. This means the labels have been converted from a single number to a vector whose length equals the number of possible classes [31]. All elements of the vector are zero except for the i^{th} element which is one and means the class is *i*. For example, the One-Hot encoded labels for the first 5 images in the test-set are:

```
In []: data.test.labels[0:5, :]
Out[]: array([[ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.],
       [ 0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.],
       [ 0.,  0.,  0.,  0.,  1.,  0.,  0.,  0.,  0.,  0.]])
```

We also need the classes as single numbers for various comparisons and performance measures, so we convert the One-Hot encoded vectors to a single number by taking the index of the highest element. Note that the word 'class' is a keyword used in Python so we need to use the name 'cls' instead.

We can now see the class for the first five images in the test-set. Compare these to the One-Hot encoded vectors above. For example, the class for the first image is 7, which corresponds to a One-Hot encoded vector where all elements are zero except for the element with index 7.

```
In []: data.test.cls[0:5]
Out[]: array([7, 2, 1, 0, 4])
```

This simple mathematical model multiplies the images in the placeholder variable x with the weights and then adds the biases. The result is a matrix of shape [*num_images, num_classes*] and weights has shape [*img_size_flat, num_classes*], so the multiplication of those two matrices is a matrix with shape [*num_images, num_classes*] and then the biases vector is added to each row of that matrix.

```
In []: logits = tf.matmul(x, weights) + biases
```

Now *logits* is a matrix with num_images rows and num_classes columns, where the element of the i'th row and j'th column is an estimate of how likely the i'th input image is to be of the j'th class.

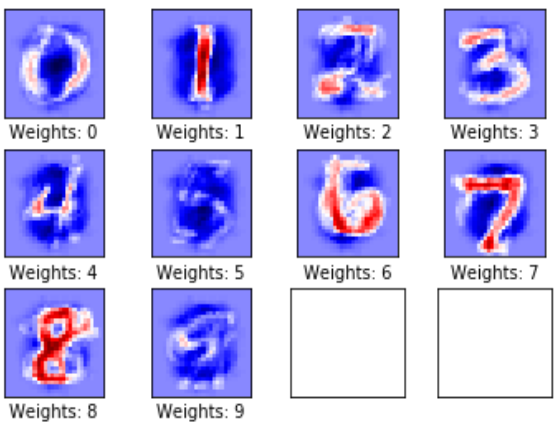
However, these estimates are a bit rough and difficult to interpret because the numbers may be very small or large, so we want to normalize them so that each row of the logits matrix sums to one, and each element is limited between zero and one. This is calculated using the so-called softmax function and the result is stored in y_pred.

```
In []: y_pred = tf.nn.softmax(logits)
```

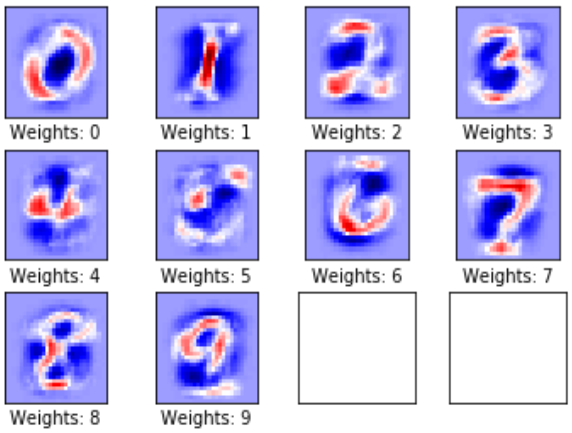
The predicted class can be calculated from the y_pred matrix by taking the index of the largest element in each row.

```
In []: y_pred_cls = tf.argmax(y_pred, axis=1)
```

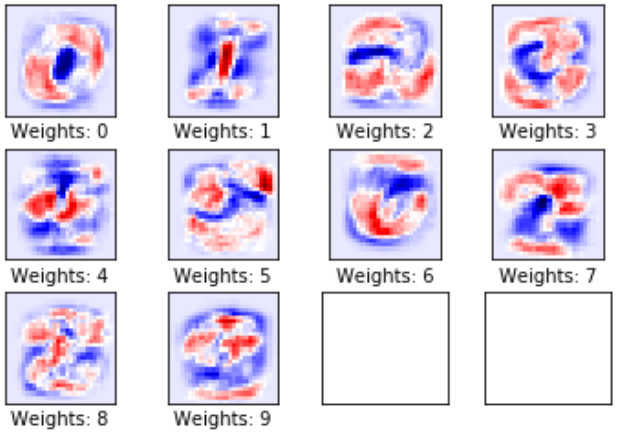
The performance before optimization gives accuracy of 9.8% on the test-set. This is because the model has only been initialized and not optimized at all, so it always predicts that the image shows a zero digit and it turns out that 9.8% of the images in the test-set happen to be zero digits. After single optimization iteration, the model increased its accuracy on the test-set to 40.7% up from 9.8%. This means that it mis-classifies the images about 6 out of 10 times. The weights can be plotted as shown in Fig 3.4 (a). Positive weights are red and negative weights are blue. These weights can be intuitively understood as image-filters. For example, the weights used to determine if an image shows a zero-digit have a positive reaction (red) to an image of a circle, and have a negative reaction (blue) to images with content in the centre of the circle [28]. Similarly, the weights used to determine if an image shows a one-digit react positively (red) to a vertical line in the centre of the image, and react negatively (blue) to images with content surrounding that line. The weights mostly look like the digits they're supposed to recognize. This is because only one optimization iteration has been performed so the weights are only trained on 100 images. After training on several thousand images, the weights become more difficult to interpret because they have to recognize many variations of how digits can be written. Similarly, after 10 optimization iteration, the model increased its accuracy on the test-set to 21.4%. The weights can be plotted as shown in Fig 3.4 (b). We can also observe and plot the confusion matrix as shown in Fig 3.5 which lets us see more details about the mis-classifications. For example, it shows that images actually depicting a 5 have sometimes been mis-classified as all other possible digits, but mostly either 3, 6 or 8.



(a) 0 Optimization Iteration



(b) 10 Optimization Iteration



(c) 1000 Optimization Iteration

Fig 3.4 Weight Plots for each digit

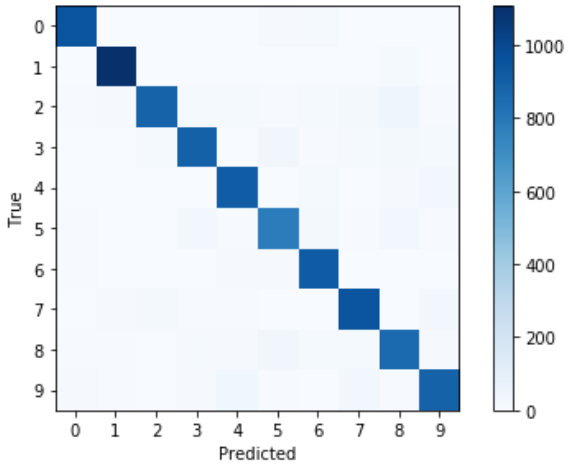


Fig 3.5 Confusion Matrix for 1000 optimization Iterations

3.2.2 Inception Model

The Inception model takes weeks to train on a monster computer with 8 Tesla K40 GPUs and probably costing $30,000 so it is impossible to train it on an ordinary PC. We will instead download the pre-trained Inception model and use it to classify images. The Inception model has nearly 25 million parameters and uses 5 billion multiply-add operations for classifying a single image. On a modern PC without a GPU this can be done in a fraction of a second per image. It is a Convolutional Neural Network with many layers and a complicated structure. The Inception model has two softmax-outputs. One is used during training of the neural network and the other is used for classifying images after training has finished, also known as inference. The Inception model is downloaded from the internet using following command:

```
In [ ]: # inception.data_dir = 'inception/'
In [ ]: inception.maybe_download()
        Downloading Inception Model ...
        Data has apparently already been downloaded and unpacked.
```

Load the Inception model so it is ready for classifying images.

```
In [ ]: model = inception.Inception()
```

Helper function is a simple wrapper-function for displaying the image, then classifying it using the Inception model and finally printing the classification scores.

```
In [ ]: def classify(image_path):
            # Display the image.
            display(Image(image_path))
            # Use the Inception model to classify the image.
            pred = model.classify(image_path=image_path)
            # Print the scores and names for the top-10 predictions.
            model.print_scores(pred=pred, k=10, only_first_name=True)4
```

The output of the Inception model is a Softmax-function. The softmax-outputs are sometimes called probabilities because they are between zero and one, and they also sum to one - just like probabilities. But they are actually not probabilities in the traditional sense of the word, because they do not come from repeated experiments. It is perhaps better to call the output values of a neural network for classification scores or ranks, because they indicate how strongly the network believes that the input image is of each possible class. Sometimes the Inception model is confused about which class an image belongs to, so none of the scores are really high. Example of this is a parrot picture with variations as explained below.

```
97.18% : macaw
0.07% : African grey
0.06% : toucan
0.05% : jacamar
0.04% : bee eater
0.03% : lorikeet
0.02% : sulphur-crested cockatoo
0.02% : jay
0.01% : kite
0.01% : indigo bunting
```

(a) Input image (b) Corresponding Output

Fig 3.6 First Test Case

In Fig 3.6, the Inception model is very confident (score about 97%) that this image shows a kind of parrot called a macaw. The Inception model works on input images that are 299×299 pixels in size. The above image of a parrot is actually 320 pixels wide and 785 pixels high, so it is resized automatically by the Inception model. In second test case as shown in Fig 3.7, there is another crop of the parrot image, this time showing its body without the head or tail. The Inception model is still very confident (score about 94%) that it shows a macaw parrot.

```
94.21% : macaw
 0.76% : toucan
 0.58% : African grey
 0.11% : jacamar
 0.10% : sulphur-crested cockatoo
 0.10% : bee eater
 0.09% : magpie
 0.08% : jay
 0.06% : lorikeet
 0.04% : hornbill
```

(a) Input image (b) Corresponding Output

Fig 3.7 Second Test Case

```
26.51% : jacamar
10.56% : grasshopper
 3.58% : chime
 2.15% : bulbul
 1.93% : fountain pen
 1.64% : leafhopper
 1.31% : kite
 1.22% : cricket
 1.09% : macaw
 0.81% : bee eater
```

(a) Input image (b) Corresponding Output

Fig 3.8 Third Test Case

In third case as shown in Fig 3.8, this image has been cropped so it only shows the tail of the parrot. Now the Inception model is quite confused and thinks the image might show a jacamar (score about 26%) which is another exotic bird, or perhaps the image shows a grass-hopper (score about 10%). The Inception model also thinks the image might show a fountain-pen (score about 2%). But this is a very low score and should be interpreted as unreliable noise.

In conclusion, the Inception model appears to have problems recognizing people and uncommon objects. This may be due to the training-set that was used. Newer versions of the Inception model have already been released, but they are probably also trained on the same data-set and may therefore also have problems recognizing people. Future models will hopefully be trained to recognize common objects such as people.

3.2.3 Transfer Learning

The Inception model is actually quite capable of extracting useful information from an image. So we can instead train the Inception model using another data-set. But it takes several weeks using a very powerful and expensive computer to fully train the Inception model on a new data-set. We can instead re-use the pre-trained Inception model and merely replace the layer that does the final classification. This is called Transfer Learning. The chart in Fig 3.9 shows how the data flows when using the Inception model for Transfer Learning. First we input and process an image with the Inception model. Just prior to the final classification layer of the Inception model, we save the Transfer Values to a cache-file. The reason for using a cache-file is that it takes a long time to process an image with the Inception model. If each image is processed more than once then we can save a lot of time by caching the transfer-values. The transfer-values are also sometimes called bottleneck-values. When all the images in the new data-set have been processed through the Inception model and the resulting transfer-values saved to a cache file, then we can use those transfer-values as the input to another neural network. We will then train the second neural network using the classes from the new data-set, so the network learns how to classify images based on the transfer-values from the Inception model. In this way, the Inception model is used to extract useful information from the images and another neural network is then used for the actual classification.

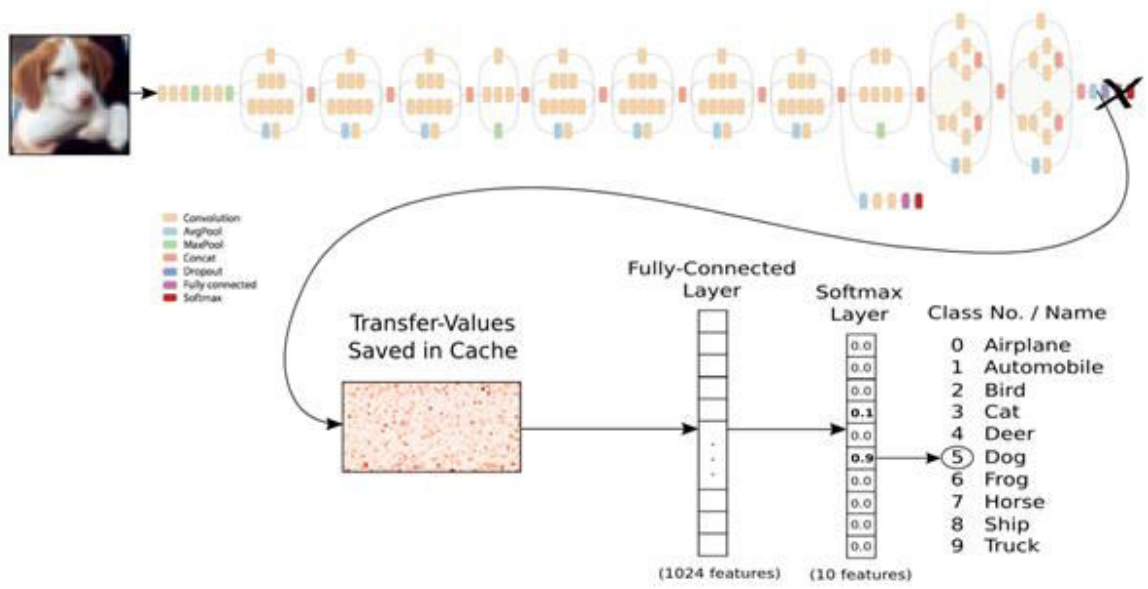


Fig 3.9 End-to-End Component Flow in Transfer Learning

It is developed using Python 3.5.2 (Anaconda) and TensorFlow version `'0.12.0-rc0'`. The data dimensions have already been defined in the cifar10 module, so we just need to import the ones we need. We begin with setting the path for storing the data-set on our computer. The data-set is about 163 MB and will be downloaded automatically if it is not located in the given path. Next step is to load the class-names followed by loading the training-set. This returns the images, the class-numbers as integers, and the class-numbers as One-Hot encoded arrays called labels as shown below:

```
In [10]:images_train, cls_train, labels_train = cifar10.
         load_training_data()
Loading data: data/CIFAR-10/cifar-10-batches-py/data_batch_1....
```

Then we load the test-set. The data-set has now been loaded and consists of 5000 images and associated labels (i.e. classifications of the images). The data-set is split into 2 mutually exclusive sub-sets, the training-set and the test-set. A Helper Function is used to plot at most 9 images in a 3x3 grid as shown in Fig 3.10, and writing the true and predicted classes below each image. The Inception model is downloaded from the internet. There is a default directory where the data-files can be saved. The directory will be created if it does not exist. Now the downloaded model is loaded so it is ready for classifying images. To calculate transfer values import a helper-function for caching the transfer-values of the Inception model.

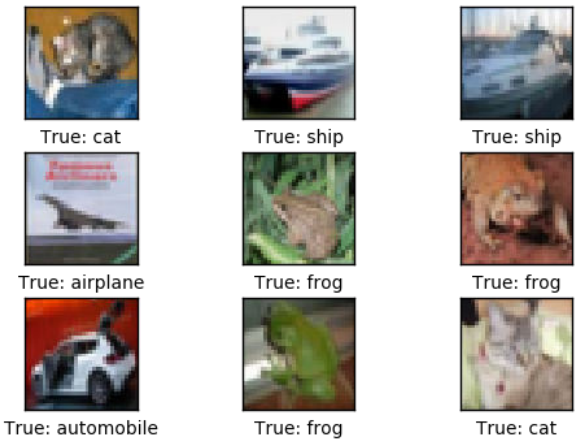
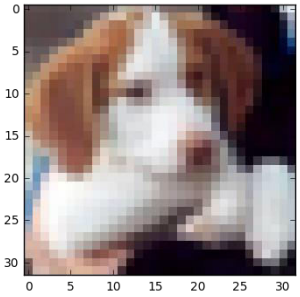
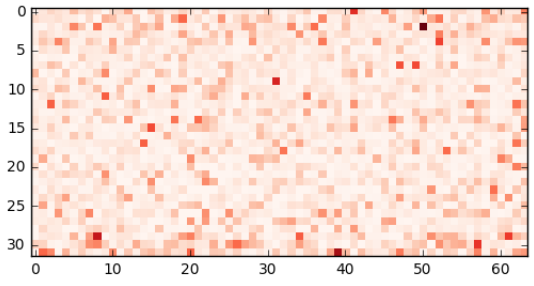


Fig 3.10 Helper function output in a 3×3 grid

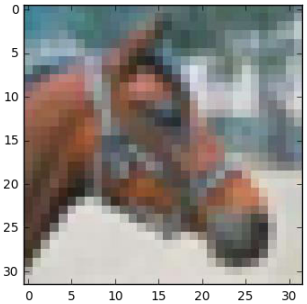
Next, set the file-paths for the caches of the training-set and test-set, check the shape of the array with the transfer-values. There are 5000 images in the training-set and for each image there are 2048 transfer-values. Similarly, there are 1000 images in the test-set with 2048 transfer-values for each image. It is used for plotting transfer values as shown in Fig 3.11.



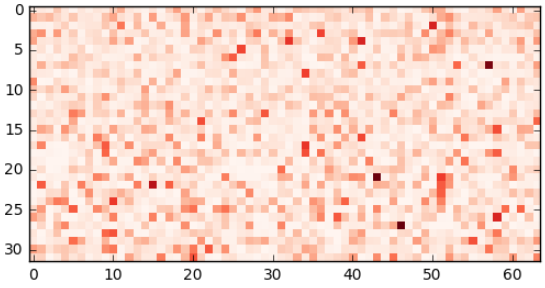
(a) Input image



(b) Corresponding plot



(a) Input image

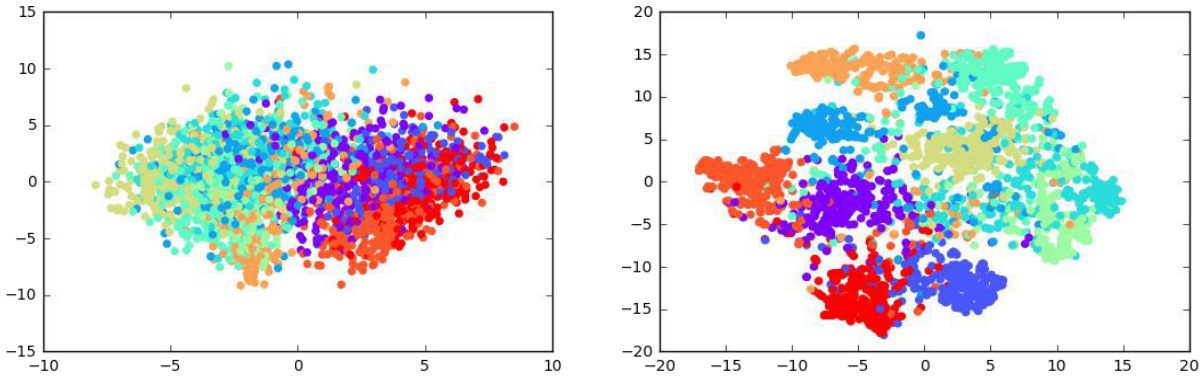


(b) Corresponding plot

Fig 3.11 Transfer-values for the image using Inception model

For analysis of Transfer-value using PCA, Use Principal Component Analysis (PCA) from scikit-learn to reduce the array-lengths of the transfer-values from 2048 to 2 so they can be

plotted. Then create a new PCA-object and set the target array-length to 2. It takes a while to compute the PCA so the number of samples has been limited to 3000. We get the class-numbers for the samples being selected. Check that the array has 3000 samples and 2048 transfer-values for each sample. Use PCA to reduce the transfer-value arrays from 2048 to 2 elements. Check that it is now an array with 3000 samples and 2 values per sample. Use Helper-function for plotting the reduced transfer-values. Plot the transfer-values that have been reduced using PCA as shown in Fig 3.12. There are 10 different colors for the different classes in the data-set. The colors are grouped together but with very large overlap. This may be because PCA cannot properly separate the transfer-values.



(a) Reduced using PCA (b) Reduced using t-SNE

Fig 3.12 Plot of transfer-values after reduction

Another method for doing dimensionality reduction is t-SNE. Unfortunately, t-SNE is very slow so we first use PCA to reduce the transfer-values from 2048 to 50 elements. We start by creating a new t-SNE object for the final dimensionality reduction and set the target to 2-dim. Then we perform the final reduction using t-SNE. We obtain an array with 3000 samples and 2 transfer-values per sample. Plot the transfer-values that have been reduced to 2-dim using t-SNE, which shows better separation than the PCA-plot above. This means the transfer-values from the Inception model appear to contain enough information to separate the images into classes, although there is still some overlap so the separation is not perfect. Finally we created another neural network in TensorFlow. This network will take as input the transfer-values from the Inception model and output the predicted classes for images.

Chapter 4

Training and Testing

After training our dataset, the next step is testing. Since our classifier is generic and can be used to classify any type of data with fair accuracy, so for the purpose of this project a casual flower datasets and, abstract art datasets whose features are inspired by psychology theory were taken for training as well as testing. The classifier was tested with multiple images from both categories, each with and without noise. Images from trained category and untrained images were picked and tested. The result gives an *evaluation time* for testing and the *probability* of the tested image belonging to each classified category in decreasing order. This chapter deals with testing of untrained images, initially. Then a set of untrained and trained images is compared by introducing multiple filters and noises. And finally results are drawn from these comparisons.

4.1 Training the Model

After preparing the dataset and categorizing the images into respective folders, training is done to teach the computer to look for. This logic is encapsulated in few lines of below shown python code:

```
SET IMAGE_SIZE=224
SET ARCHITECTURE="mobilenet_0.50_%IMAGE_SIZE%"
python -m scripts.retrain
\--bottleneck_dir=tf_files/bottlenecks
\--how_many_training_steps=500
\--model_dir=tf_files/models/
\--summaries_dir=tf_files/training_summaries/"${ARCHITECTURE}"
\--output_graph=tf_files/retrained_graph.pb
\--output_labels=tf_files/retrained_labels.txt
\ --architecture="%ARCHITECTURE%"
\ --image_dir=tf_files/Dataset_DIR
```

After running these lines, our model begins the training process which usually takes 30 to 45 minutes. The training process is by far the most important piece of the puzzle for this project. It's going to take its time to teaches the computer what there is to learn about all the images

that were curetted and its going to store them into bottleneck files which encapsulates all of the learning that has been done. The training process is shown in Fig 4.1.

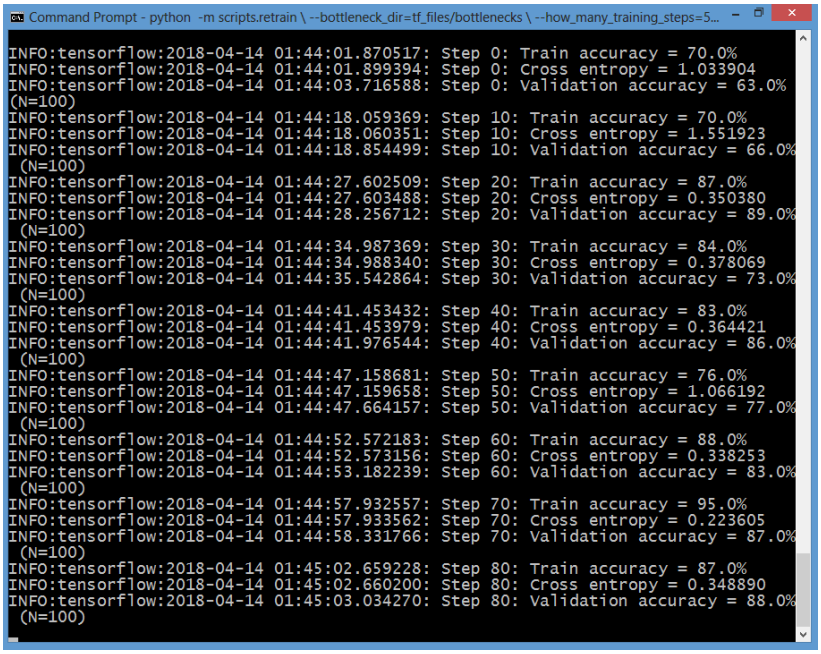


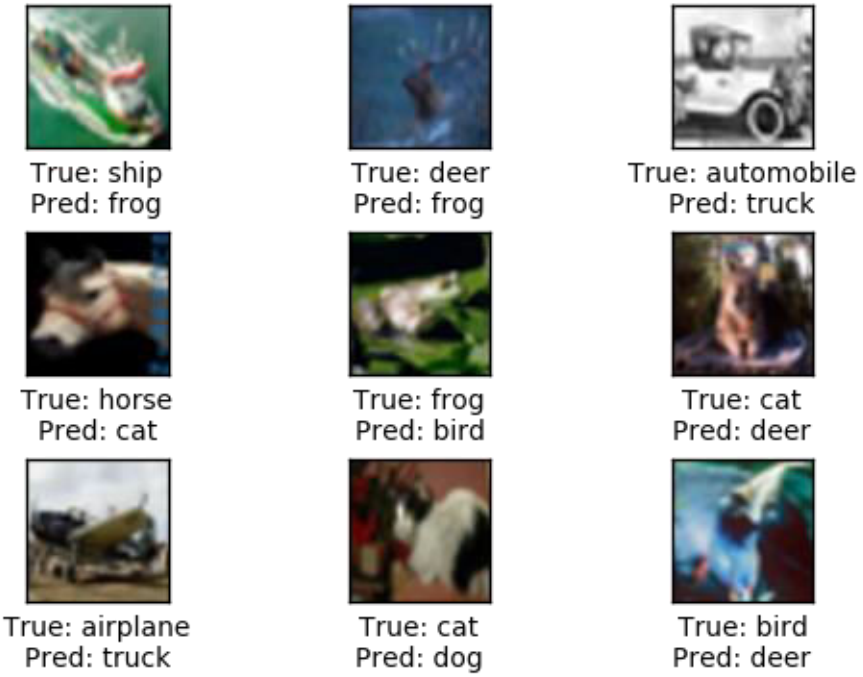
Fig 4.1 Training and storing the learned features into bottlenecks

There are 5000 images (and arrays with transfer-values for the images) in the training-set. It takes a long time to calculate the gradient of the model using all these images (transfer-values). We therefore only use a small batch of images of size 100 (transfer-values) in each iteration of the optimizer. Helper function performs a number of optimization iterations so as to gradually improve the variables of the neural network [20]. In each iteration, a new batch of data is selected from the training-set and then TensorFlow executes the optimizer using those training samples [21]. The progress is printed every 100 iterations. This function also calculates the predicted classes of images and returns a boolean array whether the classification of each image is correct. It calculates the classification accuracy given a boolean array whether each image was correctly classified. For example, `classification accuracy([True, True, False, False, False]) = 2/5 = 0.4`. The function also returns the number of correct classifications. The classification accuracy on the test-set is very low because the model variables have only been initialized and not optimized at all, so it just classifies the images randomly.

`Accuracy on Test-Set: 9.4% (939 / 10000)`

After 10,000 optimization iterations, the classification accuracy is about 90% on the test-set which is better when compared to the basic Convolutional Neural Network which had less than 80% accuracy on the test-set.

```
Global Step:    100, Training Batch Accuracy:  82.8%
Global Step:    200, Training Batch Accuracy:  90.6%
Global Step:    300, Training Batch Accuracy:  90.6%
Global Step:    400, Training Batch Accuracy:  95.3%
Global Step:    500, Training Batch Accuracy:  85.9%
Global Step:    600, Training Batch Accuracy:  84.4%
Global Step:    700, Training Batch Accuracy:  90.6%
.........................
Accuracy on Test-Set: 90.7% (9069 / 10000)
Example errors:
```




```
Confusion Matrix:
[926   6  13   2   3   0   1   1  29  19] (0) airplane
[  9 921   2   5   0   1   1   1   2  58] (1) automobile
[ 18   1 883  31  32   4  22   5   1   3] (2) bird
[  7   2  19 855  23  57  24   9   2   2] (3) cat
[  5   0  21  25 896   4  24  22   2   1] (4) deer
[  2   0  12  97  18 843  10  15   1   2] (5) dog
[  2   1  16  17  17   4 940   1   2   0] (6) frog
[  8   0  10  19  28  14   1 914   2   4] (7) horse
[ 42   6   1   4   1   0   2   0 932  12] (8) ship
[  6  19   2   2   1   0   1   1   9 959] (9) truck
  (0) (1) (2) (3) (4) (5) (6) (7) (8) (9)
```

4.2 Testing Images (Without Noise)

Untrained images are the images which are not directly picked from the dataset used for training and are new to the algorithm. The task of algorithm is to predict and classify it into one of the categories. They are images which may or may not belong one of the categories of classifier and but were specifically not used for training the classifier.

4.2.1 Testing of flower dataset

(a)

```
Evaluation time (1-image): 0.733s

dandelion 0.96812123
daisy 0.021511387
sunflowers 0.005121417
roses 0.004219075
tulips 0.0010268184
```

(b)

```
Evaluation time (1-image): 0.769s

tulips 0.9324618
daisy 0.06683462
roses 0.00039041272
sunflowers 0.0001622207
dandelion 0.00015093258
```

(c)

```
Evaluation time (1-image): 0.747s

daisy 0.5650241
dandelion 0.28550655
sunflowers 0.1436033
roses 0.004945735
tulips 0.00092028256
```

(d)

```
Evaluation time (1-image): 0.856s

roses 0.6585604
tulips 0.3413031
daisy 0.00013549274
dandelion 6.5375514e-07
sunflowers 3.7700147e-07
```

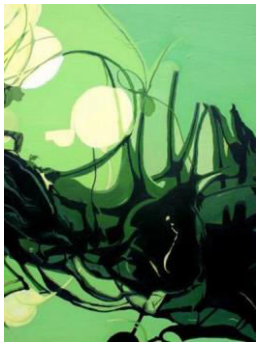
Fig 4.2 Test Images and their Corresponding Results

After training our dataset for flower classification, the model was tested with 500 test images. Fig 4.2 show four examples of test images and their corresponding result. Fig 4.2 (a) shows a picture of a dandelion with full sun view in background, though it can be easily misunderstood as sunflower but our model classified it correctly with ~96.81% probability of it being a dandelion. Fig 4.2 (d) is not a single flower but is collection of roses and tulips. It can be seen that the quantity is noticeably less than roses which is correctly predicted by our classifier. Classifier predicts ~65.85% roses and 34.13% tulips. In conclusion, our model classifies ~87 images precisely in every 100 images.

4.2.2 Testing of Abstract Art Dataset for Emotion Prediction

The selection and development of useful image features is an open research topic. For each application, different features are needed to fulfil the task at hand. We introduce features based on the experimentally-determined relation between color saturation and brightness, and emotion dimensions, as well as features based on relations between color combinations and induced emotional effects from art theory [19]. We complement these features by a selection of features, some of which are shown to be of use in similar image retrieval and classification tasks. In this work, features representing the color, texture, composition and content were implemented. For testing and training, we use three data sets: (1) the International Affective Picture System (IAPS), (2) a set of 807 artistic photographs downloaded from an art sharing site, (3) a set of 228 peer rated abstract paintings. The fourth data set is a combined set of all of the above.

(a)



```
Evaluation time (1-image): 0.818s

sad 0.71000665
excitement 0.28997043
fear 2.0529982e-05
amusement 2.1126284e-06
contentment 1.16909945e-07
```

(b)

```
Evaluation time (1-image): 0.785s

excitement 0.99981385
fear 7.65642e-05
sad 6.329158e-05
anger 4.478041e-05
contentment 1.4354254e-06
```

(c)

```
Evaluation time (1-image): 0.798s

anger 0.99914527
sad 0.00063045893
disgust 0.00020110104
awe 2.2465003e-05
excitement 5.095727e-07
```

(d)

```
Evaluation time (1-image): 0.950s

excitement 0.6371615
contentment 0.33817327
awe 0.02080013
fear 0.0022090953
disgust 0.00087178807
```

Fig 4.3 Test Image and their corresponding Results

Table 4.1 shows our resulting best feature selection for each data set and each category, along with the dimensionality reduction method producing each feature set listed in the last row. It can also be seen the images sets are unbalanced. As shown in Fig 4.3, for the IAPS set, the Yanulevskaya et al. features give the best performance for the Anger and Contentment classes. For these classes [18], we show the best selection from the features implemented in this study. It is clear that the best feature set is dependent on both the category and the data set. The occurrence and size of human faces was clearly the strongest feature for the Amusement category of the IAPS image set. The categories in the IAPS set are strongly content related, e.g. Fear and Disgust images often show snakes, insects or injuries, whereas Amusement images often contain portraits of happy people [16]. The classifier has the best performance when it basically detects the portraits. However, there is no such strong connection between faces and categories in the artistic sets. Instead the colors become much more important for the artistic photos. We see that features based on art theories (Itten colors, Wang Wei-ning histograms) are indeed most often selected for the artistic photos set. A large number of Itten features are selected for the Amusement and Excitement classes [17]. The color features developed by Wang Wei-ning et al. are also effective for this task- these features on their own had the best performance for the Awe and Disgust classes, and were also selected by the feature selection algorithms for the Amusement and Excitement classes.

Table 4.1 Number of images per data set and emotion category

	Amusement	Anger	Awe	Contentment	Disgust	Excitement	Fear	Sad	sum
IAPS	37	8	54	63	74	55	42	61	394
Art photo	101	77	103	70	70	105	115	166	807
Abstract paintings	25	3	15	63	18	36	36	32	228
Combined	163	88	172	196	162	196	193	259	1429

4.3 Testing Images after Introducing Noise

The same testing sets were introduced with four different noises – Gaussian, Salt and paper, Speckle and Poisson. The Flower dataset did not give much difference in result even after introduction of noise but in Abstract Art dataset, the classifier switched the categories in different noises.

4.3.1 Testing of Flower Data Set

(a)

```
Evaluation time (1-image): 0.712s

dandelion 0.77962667
daisy 0.17675899
roses 0.034471497
tulips 0.008377665
sunflowers 0.0007651793
```

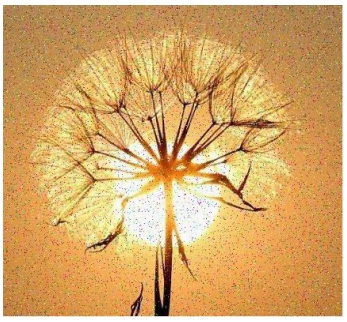
(b)

```
Evaluation time (1-image): 0.695s

dandelion 0.9448079
daisy 0.03846984
roses 0.012356132
tulips 0.0035570727
sunflowers 0.00080899475
```

(c)

```
Evaluation time (1-image): 0.875s

dandelion 0.92076457
daisy 0.04083921
roses 0.01973606
tulips 0.017444352
sunflowers 0.0012158485
```

(d)



```
Evaluation time (1-image): 0.797s

dandelion 0.92114806
roses 0.044124667
daisy 0.024325186
tulips 0.007314661
sunflowers 0.0030874342
```

Fig 4.4 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

(a)

```
Evaluation time (1-image): 0.750s

roses 0.67337817
tulips 0.2755528
sunflowers 0.037467863
daisy 0.013290667
dandelion 0.00031046045
```

(b)

```
Evaluation time (1-image): 0.734s

tulips 0.8013231
roses 0.16572559
sunflowers 0.020760095
daisy 0.011045441
dandelion 0.0011457915
```

(c)

```
Evaluation time (1-image): 0.720s

tulips 0.5187396
roses 0.4426647
sunflowers 0.031984884
daisy 0.0058029876
dandelion 0.00080790557
```

(d)

```
Evaluation time (1-image): 0.719s

roses 0.9931607
tulips 0.00510673
sunflowers 0.0016324222
daisy 7.494997e-05
dandelion 2.5210347e-05
```

Fig 4.5 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

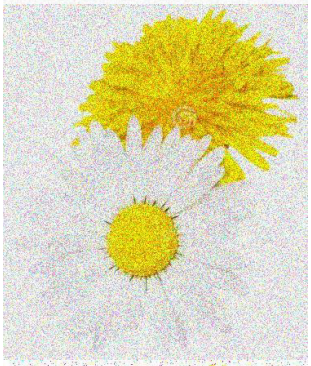
(a)

```
Evaluation time (1-image): 0.828s

dandelion 0.563459
sunflowers 0.4031017
daisy 0.033215668
roses 0.00014555008
tulips 7.8047815e-05
```

(b)

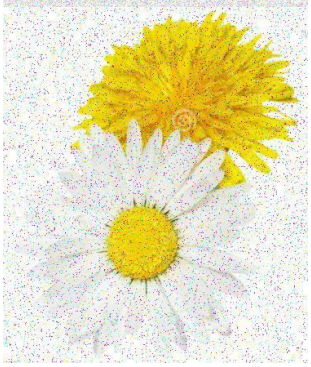
```
Evaluation time (1-image): 0.781s

dandelion 0.6714398
sunflowers 0.19444655
daisy 0.13371629
roses 0.000231695
tulips 0.0001656952
```

(c)



```
Evaluation time (1-image): 0.797s

sunflowers 0.51665854
dandelion 0.44171837
daisy 0.037761893
roses 0.002782922
tulips 0.0010782094
```

(d)



```
Evaluation time (1-image): 0.797s

sunflowers 0.51665854
dandelion 0.44171837
daisy 0.037761893
roses 0.002782922
tulips 0.0010782094
```

Fig 4.6 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

(a)

```
Evaluation time (1-image): 0.734s

roses 0.9740117
tulips 0.025980506
daisy 4.045147e-06
sunflowers 3.2365342e-06
dandelion 4.917805e-07
```

(b)

```
Evaluation time (1-image): 0.703s

roses 0.83941567
tulips 0.1605834
sunflowers 5.213814e-07
daisy 3.5684548e-07
dandelion 1.3404562e-08
```

(c)

```
Evaluation time (1-image): 0.734s

roses 0.9529998
tulips 0.04676839
sunflowers 0.00019749002
daisy 3.332606e-05
dandelion 1.0741289e-06
```

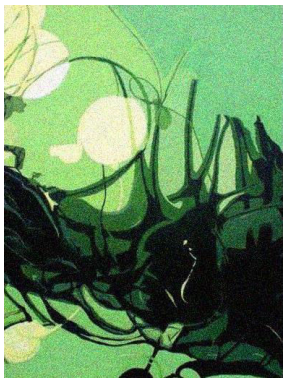
(d)

```
Evaluation time (1-image): 0.703s

roses 0.9907478
tulips 0.009247784
sunflowers 2.0644761e-06
dandelion 1.2852237e-06
daisy 1.1557656e-06
```

Fig 4.7 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

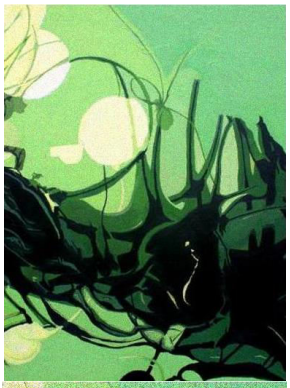
4.3.2 Testing of Abstract Art Dataset for Emotion Prediction

(a)



```
Evaluation time (1-image): 0.688s

excitement 0.5481936
sad 0.2653743
disgust 0.17750067
fear 0.008906354
contentment 2.0170317e-05
```

(b)

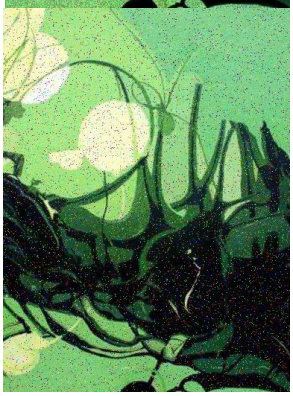


```
Evaluation time (1-image): 0.797s

excitement 0.55620795
sad 0.3157653
disgust 0.12680633
fear 0.0010658583
amusement 0.00013706881
```

(c)

```
Evaluation time (1-image): 0.703s

disgust 0.8956589
excitement 0.05989515
sad 0.0443632
fear 7.598538e-05
amusement 6.490693e-06
```

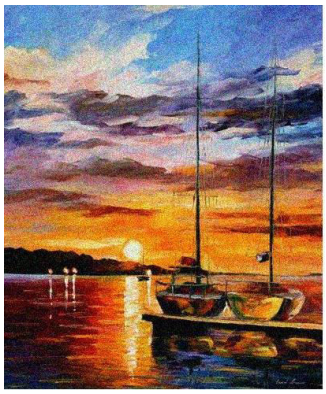
(d)



```
Evaluation time (1-image): 0.734s

disgust 0.9954745
excitement 0.0024862925
sad 0.0017729072
fear 0.0002651375
awe 7.631548e-07
```

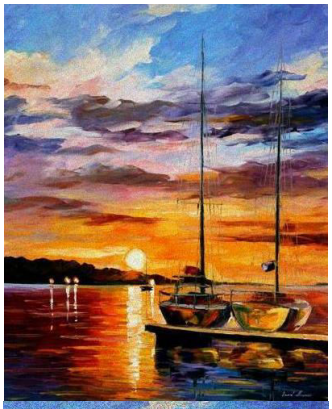
Fig 4.8 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

(a)



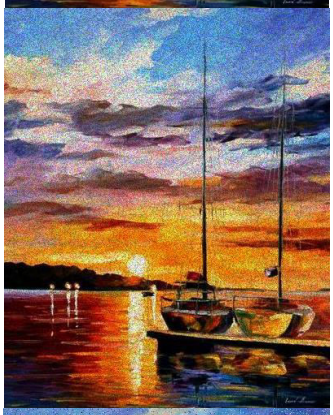
```
Evaluation time (1-image): 0.766s

contentment 0.92441726
excitement 0.054525692
disgust 0.010763442
fear 0.010004607
awe 0.00023073751
```

(b)



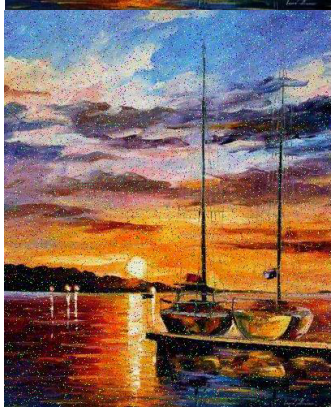
```
Evaluation time (1-image): 0.813s

contentment 0.8407954
excitement 0.14962725
disgust 0.005892289
fear 0.0022130478
awe 0.0013461664
```

(c)



```
Evaluation time (1-image): 0.781s

contentment 0.9769571
excitement 0.010847451
disgust 0.010215575
fear 0.0016369855
sad 0.0001578044
```

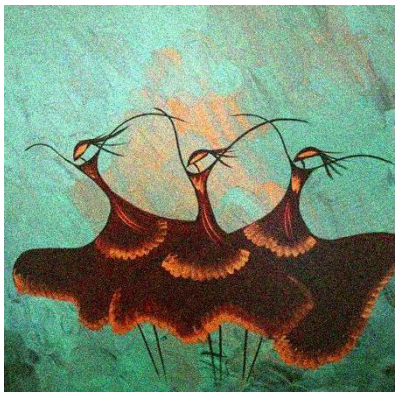
(d)



```
Evaluation time (1-image): 0.813s

contentment 0.9942352
excitement 0.0023747182
disgust 0.001970696
fear 0.0013299286
sad 7.4406205e-05
```

Fig 4.9 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

(a)



```
Evaluation time (1-image): 0.797s

excitement 0.8842932
sad 0.11433454
disgust 0.0006457548
anger 0.00061827793
amusement 7.455675e-05
```

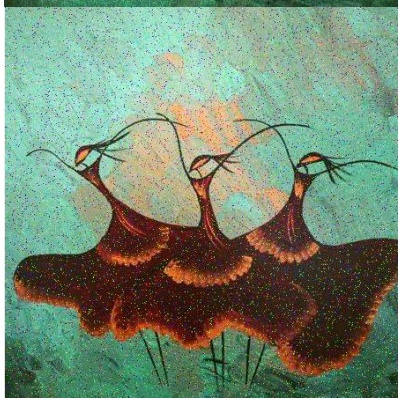
(b)

```
Evaluation time (1-image): 0.750s

excitement 0.9731214
sad 0.02624241
disgust 0.00058867555
anger 2.393661e-05
amusement 1.5142615e-05
```

(c)

```
Evaluation time (1-image): 0.720s

excitement 0.99935395
sad 0.0003927086
anger 0.00022271533
contentment 2.623326e-05
disgust 3.1006862e-06
```

(d)



```
Evaluation time (1-image): 0.906s

sad 0.9781246
excitement 0.0146134
disgust 0.0071695484
contentment 5.1065712e-05
amusement 3.217607e-05
```

Fig 4.10 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

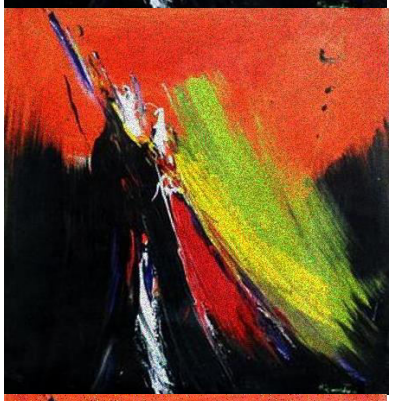
(a)

```
Evaluation time (1-image): 0.703s

sad 0.57629097
anger 0.40255734
disgust 0.013804278
amusement 0.0026949528
excitement 0.0025886798
```

(b)

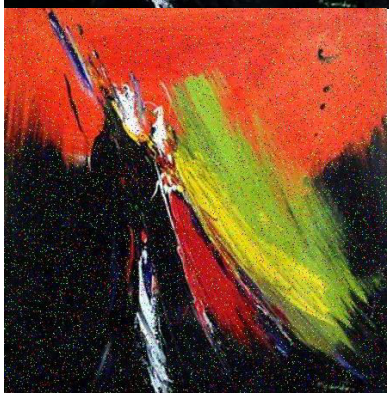
```
Evaluation time (1-image): 0.703s

sad 0.6810754
anger 0.31525332
amusement 0.0027905984
awe 0.00065452326
disgust 0.00018611512
```

(c)



```
Evaluation time (1-image): 0.703s

sad 0.9706762
anger 0.0239843
amusement 0.002771465
disgust 0.0018973522
excitement 0.0005357631
```

(d)



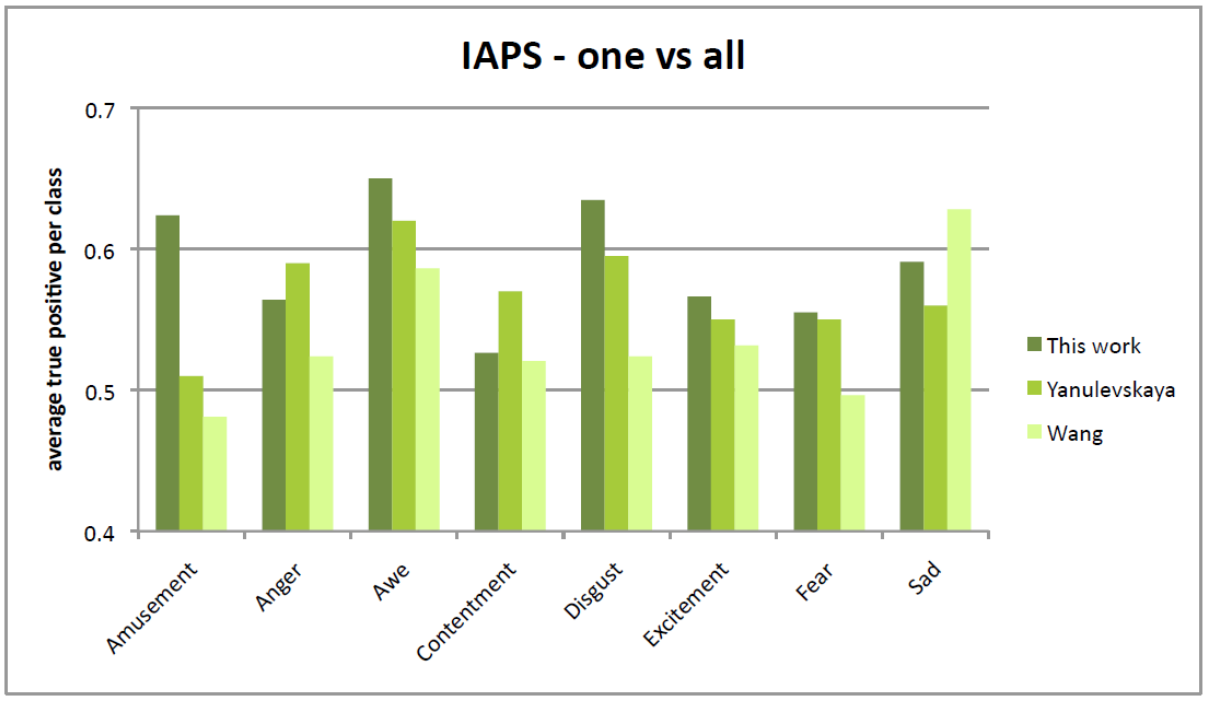
```
Evaluation time (1-image): 0.688s

sad 0.8692923
disgust 0.071787134
anger 0.04528664
excitement 0.01345951
amusement 8.6549284e-05
```

Fig 4.11 (a) Test image with Gaussian Noise and its corresponding result (b) Poisson Noise (c) Speckle Noise (d) Salt and Pepper Noise

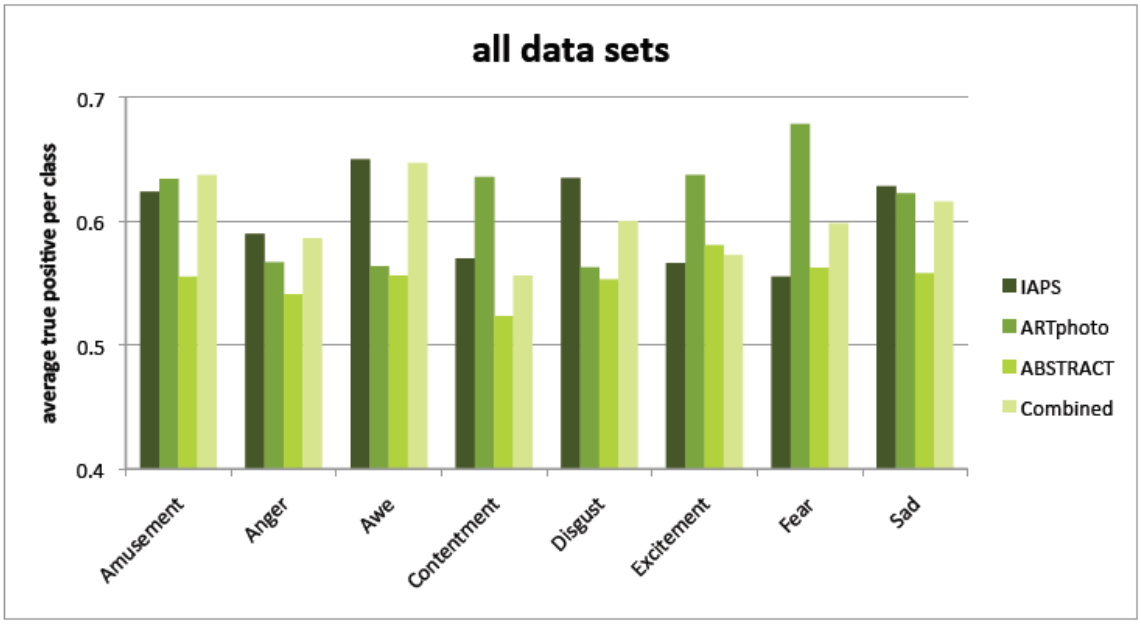
In Graph 4.1 the results for the IAPS are shown, taking the best performing features for each class, and comparing them to the best performing features. As can be deduced from the diagram, our feature sets outperform both the results by Yanulevskaya [13] and the Wang Wei-ning [14] features on the IAPS data set for five of eight categories. This shows that features created specifically for the task of affective image classification perform well, outperforming the more generic features of Yanulevskaya et al. [15] for the majority of the classes. Graph 4.2 shows classification results for the best features for every data set used in this work. It shows no clear difference in results for the IAPS and the artistic photo image sets. For half of the classes the results are better for the IAPS set, for the other half for the

artistic photo set. The classification of the abstract paintings data set has the worst performance. In Graph 4.3 we compare the performance of our best performing features for each class for the abstract paintings data set with the results achieved with the features, but also with humans, based on the multiple responses in the web survey. The bars labelled "Human" in Graph 4.3 where calculated by taking the number of votes for the category with the most votes and dividing it by the overall number of votes that the picture received. This gives us a measure of agreement among the survey participants, which can be interpreted as confidence in the winning emotion category. There is generally good correlation between the level of agreement of the humans and the results of the proposed algorithm. The largest difference is in the anger class, but this is likely due to the small number of images in this class, which makes the classification results less reliable.

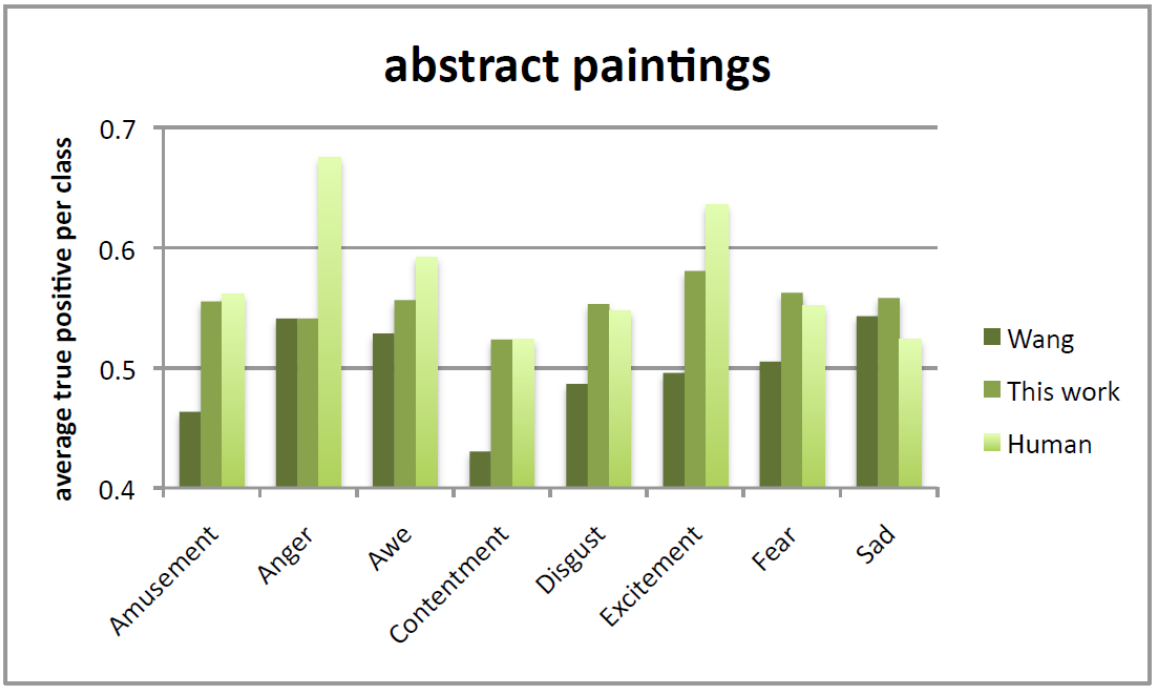
Graph 4.1 Classification performances for IAPS taking our best features for each category compared against the best features from Yanulevskaya and the feature set described in Wang



Graph 4.2 Classification performance for all image sets used in this work. The results are from the best feature selections implemented during this work.



Graph 4.3 Classification performance of the abstract paintings image set taking our best features for each category.



Chapter 5

Conclusion and Future Scope

In this project, we used the inception model theory and transfer model to achieve a classification accuracy of about ~87% on the flower dataset. This was done by feeding all the images from the casual flower dataset through the Inception model and caching the transfer-values prior to the final classification layer, hence, making full use of transfer learning. We then built another neural network that took these transfer-values as input produced from an abstract art dataset which gave us improved classification results on the International Affective Picture System (IAPS), compared to state of the art work as output.

The data-set contains a total of 5000 images. It took about 3 hours to calculate the transfer-values of the Inception model for all these images, using a laptop PC without a GPU. And training the new classifier on top of these transfer-values only took a few minutes. So the combined time-usage of this transfer-learning was less than half the time it took to train a neural network directly for the data-set, and it achieved significantly higher classification accuracy.

For evaluation, we conducted experiments to measure the performance of our features and compared the results with that of Yanulevskaya et al. [13] and also to Wang Weining et al [14]. The experimental setup was as follows: each category was separated against all others in turn, in other words a "one category against all" setup. We separate the data into a training and test set using K-fold Cross Validation (K = 5). Since we do not have a balanced data set in terms of the number of examples per category, but the probabilities for the categories should be the same for all, we optimize for the true positive rate per class averaged over the positive and negative classes, instead of the correct rate over all samples. This procedure is independent of the number of positive and negative samples. Hence we do not have to sub-sample the classes. This measure is also used as the evaluation measure for the experimental results. In the IAPS picture set was used for evaluation, as well as the same emotional categories as in this work, so the results are directly comparable. Similar to this, we perform

dimensionality reduction on the feature vectors, for which we use two algorithms. Two feature selection algorithms are used: a wrapper-based subset evaluation with a genetic search algorithm (referred to as the wrapper-based method), and an algorithm based on the classification performance of a single feature at a time, selecting only those features which resulted in an average true positive rate per class higher than 0.51 (referred to as the single feature method). We also use a feature extraction algorithm, principal component analysis (PCA). Results using all features were also computed. Following Yanulevskaya's approach we select our best feature subsets for each category (as the best performing subset from the above feature selection and extraction algorithms) and compare their performance to the best results in Yanulevskaya et al. Wang Wei-ning et al. , in contrast, used an unknown data set as well as different categories. To get comparable results, we re-implemented the features from it (further referred to as "Wang Histogram") and used them in our classification scheme. So transfer-learning with the Inception model is useful for building an image classifier for our own data-set.

We used a selection of features specific to the problem of affective image classification and achieved results that are better than comparable state of the art. There is nevertheless potential for improvement, both in developing better features and better classification. Especially semantic-based features, such as the recognition of the emotional expression of faces, or certain common symbols (e.g. hearts) could be of advantage. For the classification, rather than forcing each image to be in a single emotional category, an "emotional histogram" showing a distribution over the categories could be produced. Furthermore, more reliable ground truth from a larger number of people is required- this will set an upper limit on the classification accuracy that should be achieved by such approaches. A further potential development is learning the preferences of individual people, instead of the consensus based approach adopted in this work.

References

[1] Noridayu Manshor and Dhanesh Ramachandram, "Feature Fusion in Improving Object Class Recognition," Journal Computer Sci, 2012.

[2] Hideki Nakayama,Tatsuya Harada and Yasuo Kuniyoshi, "Scene Classification using Generalized Local Correlation," IAPR Conference on Machine Vision Applications, May 20-22, 2009.

[3] J. Vogel and B. Schiele, "Semantic modeling of natural scenes for content-based image retrieval," Int. J. Comput.Vis.

[4] S. Daniel Madan Raja, Dr.A.Shanmugam, "ANN and SVM Based War Scene Classification using Wavelet Features: A Comparative Study," Journal of Computational Information Systems, pp. 1402-1411–86, 2011.

[5] Demir Gokalp and Selim Aksoy, "Scene Classification Using Bag-of-Regions Representations," 2005.

[6] L. J. Li and L. Fei-Fei, "What, where and who? Classifying event by scene and object recognition," IEEE Int. Conf. Comput. Vis., pp. 1–8, Feb 2007.

[7] L.Fei-Fei and P. Perona, "A Bayesian hierarchical model for learning natural scene categories," IEEE Conf. Comput. Vis. PatternRecognit., pp. 524-531, March 2005.

[8] Le Dong and Jiang Su, "Scene-Oriented Hierarchical Classification of Blurry and Noisy Images," IEEE Trans, vol. 21, no. 5, pp. 712–727, May 2012.

[9] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," IEEE Conf. Comput. Vis. Pattern Recognit., vol. 2, pp. 2169–2178, March 2006.

[10] Jipsa Kurian, V.Karunakaran," A Survey on Image Classification Methods" ISSN: 2278 – 909X International Journal of Advanced Research in Electronics and Communication Engineering (IJARECE) Volume 1, Issue 4, October 2012.

[11] Andreopoulos, Alexander & Tsotsos, John "50 Years of object recognition: Directions forward. Computer Vision and Image Understanding". 117, 827–891, April 2013.

[12] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi University of Washington, Allen Institute for AI, Facebook AI Research "You Only Look Once: Unified, Real-Time Object Detection".

[13] V. Yanulevskaya, J. C. van Gemert, K. Roth, A. K. Herbold, N. Sebe, and J. M. Geusebroek. Emotional valence categorization using holistic image features. In IEEE Int. Conf. on Image Processing, 2008.

[14] W. Wei-ning, Y. Ying-lin, and J. Sheng-ming. Image retrieval by emotional semantics: A study of emotional space and feature extraction. IEEE Int. Conf. on Systems, Man and Cybernetics, 4(Issue 8-11):3534-3539, Oct. 2006.

[15] J. M. Corridoni, A. Del Bimbo, and P. Pala. Image retrieval by color semantics. Multimedia Syst., 7(3):175-183, 1999.

[16] R. Datta, D. Joshi, J. Li, and J. Z. Wang. Studying aesthetics in photographic images using a computational approach. In ECCV (3), pages 288-301, 2006.

[17] I. Daubechies. Ten Lectures on Wavelets. Regional Conf. Series in Applied Mathematics. Soc for Industrial & Applied Math, December 1992.

[18] P. Ekman, W. V. Friesen, M. O'Sullivan, A. Chan, I. Diacoyanni-Tarlatzis, K. Heider, R. Krause, W. A. LeCompte, T. Pitcairn, P. E. Ricci-Bitti, K. Scherer,

[19] M. Tomita, and A. Tzavaras. Universals and cultural differences in the judgments of facial expressions of emotion. Journal of Personality and Social Psychology, 53(Issue 4):712-717, Oct 1987.

[20] A. Hanbury. Constructing cylindrical coordinate colour spaces. Pat. Rec. Lett., 29(4):494-500, 2008.

[21] A. Hanjalic. Extracting moods from pictures and sounds: towards truly personalized TV. Signal Processing Magazine, IEEE, 23(2):90-100, 2006.

[22] A. Hanjalic and L.-Q. Xu. Affective video content representation and modeling. IEEE Transactions on Multimedia, 7(1):143-154, 2005.

[23] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Insideoutside net: Detecting objects in context with skip pooling and recurrent neural networks. arXiv preprint arXiv:1512.04143, 2015.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. FeiFei.Imagenet: A large-scale hierarchical image database. In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, pages 248–255. IEEE, 2009.

[25] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2):303–338, 2010.

[26] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Discriminatively trained deformable part models, release 4. http://people.cs.uchicago.edu/ pff/latent-release4/.

[27] R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385, 2015.

[29] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems, pages 1097–1105, 2012.

[31] M. Lin, Q. Chen, and S. Yan. Network in network. arXiv preprint arXiv:1312.4400, 2013.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick. Microsoft coco: Common objects in context. In European Conference on Computer Vision, pages 740–755. Springer, 2014.

[33] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. SSD: single shot multibox detector. CoRR, abs/1512.02325, 2015.

[34] G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. J. Miller. Introduction to wordnet: An on-line lexical database. International journal of lexicography, 3(4):235–244, 1990.

[35] Jana Machajdik, Allan Hanbury. Affective Image Classification using Features Inspired by Psychology and Art Theory. In European Conference on Computer Vision, pages 740–755. Springer, 2013. MM'10, October 25–29, 2010, Firenze, Italy.

Plagirism Report

Article-321

ORIGINALITY REPORT

16%	3%	10%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.arxiv.org/pdf/1506.02640.pdf Publication	3%
2	citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.683.210&rep=rep1&type=pdf Publication	2%
3	www.arxiv.org/ftp/arxiv/papers/1702/1702.00723.pdf Publication	2%
4	www.imageemotion.org Internet Source	2%
5	Submitted to University of Durham Student Paper	1%
6	docs.grafana.org Student Paper	1%
7	Submitted to CSU, San Jose State University Student Paper	1%

8	www.portanna.org Internet Source	1%
9	Submitted to University of Durham Publication	1%
10	docs.grafana.org Publication	1%
11	Submitted to CSU, San Jose State University Publication	1%

Exclude quotes	On	Exclude matches	< 1%
Exclude bibliography	On		

Application of machine learning algorithms for profile reconstruction of IPM device

Measured IPM profiles can be significantly distorted due to the displacement of residual ions or electrons by interaction with beam fields for high brightness or high energy beams [1, 2, 3]. It is thus difficult to deduce the characteristics of the actual beam from the measurements. In this project different Machine Learning Regression Algorithms are applied to reconstruct the actual beam profile from the measurement data.

1 Introduction

Ionization Profile Monitors (IPM) are used for nondestructive transverse beam profile measurements at many accelerator facilities. The principle of operation is the following; the primary beam ionizes the residual gas and the ionized particles (ions or electrons) are extracted via electric fields and sometimes in conjunction with magnetic fields to confine the movement of ionized particles in the plane transverse to the electric field. The profile of the extracted particles reflects the transverse profile of the primary beam with the assumption that ionized particles are created at rest and the effect of induced fields by the primary beam on ionized particles can be neglected.

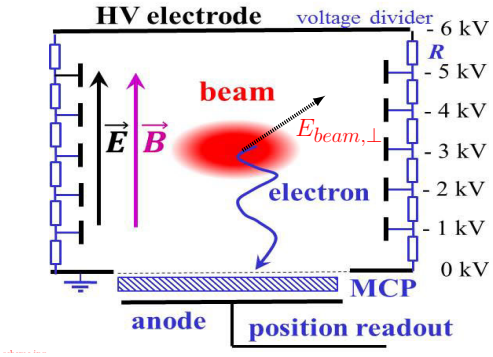


Fig. 1: *Operating principle of the IPM.*

Figure 1 shows the typical components present in an IPM where both the electric and magnetic fields are utilized to confine the ionized particles [6]. The support electrodes/rods between the top and bottom electrodes are used to reduce the fringe fields and improve field homogeneity. The field homogeneity is important in order to avoid any distortion in the measured profile and therefore static EM simulations for the full geometry are usually performed. IPMs are often used for nondestructive measurements in low-pressure conditions such as storage rings and hence they usually have to be equipped with a high amplification multichannel plate (MCP) for obtaining sufficient signal to noise ratio. The outputs of MCPs are connected to data acquisition system directly or via phosphor screens and an optical system.

The distortion in measured IPM profiles due to beam space charge is well known and typically magnetic fields are used to confine the generated ionized particles around the point of generation. However, in some cases required magnetic field strengths are prohibitory for extremely dense beams as discussed below for our target case. First major distortion for an IPM with magnetic fields was seen for LHC IPMs at high energies, where the beam profile was significantly broader compared to wire scanner measurements [1]. The first solution envisaged to solve the issue was to raise the magnetic fields in the IPM to 1 T [2]. This field strength should be viewed in perspective of the strength of LHC main dipoles which is 8.5 T at top energies and was considered impractical. Recently reliable IPM simulation tools have been developed as a joint effort between several labs [5]. Availability of reliable description of IPM system including the profile distorting effects such as space charge and initial velocity distribution of ionized particles transforms the problem of IPM profile correction into a "supervised learning" problem. In a supervised learning problem, the input and output of an unknown system are provided and the system is approximated by a variety of machine learning algorithms [7]. Here we will apply

several Machine Learning (ML) regression algorithms to reconstruct actual beam profile from the measured distorted profile. In the next section, the simulation tool, as well as the beam and device parameters used to train the ML regression models, are discussed. Following that, the ML algorithms, their respective parameters, training, and validation are presented and the results are summarized.

2 Machine Learning Regression

The ability to find the underlying process model and its features from data are referred to as machine learning. This can be used to predict the system behavior or make decisions. The field is evolved primarily from pattern recognition and Artificial Intelligence. Though Machine Learning (ML) algorithms have a variety of applications such regression, classification, dimensionality reduction, clustering, density estimation, we focus on regression/prediction in our context. The choice of ML algorithm is problem specific, and typical deciding factors are bias-variance tradeoff, training time, and dimensionality of the problem (i.e.the number of inputs and outputs), and any prior information about the features of the underlying model.

Each Machine Learning (ML) regression algorithms involve three important functions:

"Machine Learning = Representation + Evaluation + Optimization"

A) Mathematical learning model (Representation)/Decision function: It is an approximation of the underlying function between inputs and outputs which is later used for predictions of the unknown input dataset.

B) Loss function/Error function (Evaluation): Function indicating the $L : y \times y \rightarrow \mathbb{R}_+$ penalty for an incorrect prediction [19]. It is used for finding error percentage between predicted output and true output. Evaluation is essentially how you judge or prefer one model vs. another.

C) Optimization Function(Optimization): it is used to search optimal parameters for mathematical learner model. In most algorithms "gradient descent method" is used as optimization function. This is how you search the space of represented models to obtain better evaluations.

Four typical ML regression methods with increasing level of complexity are applied to the problem as discussed below:

1) Linear Regression: It is a linear approach for modeling the relationship between a scalar dependent variable y and one or more explanatory variables denoted X. The Mathematical learning model (Representation) of linear regression can be represented by,

$$y_{pred} = WX + b \tag{1}$$

where W is an array of coefficients and X(X_1,X_2. X_n) is a $M \times N$ matrix where M is the number of data sets, N is the number of features of each data set. b is bias, predicted output is y_{pred}.Predicted output y_{pred} is an array of length M. whereas the equation of loss function of linear regression is [12],

$$min \sum (y_{real} - y_{pred})^2 \tag{2}$$

where y_{real} is true output values for specific input value X.

2) Ridge Regression: To avoid mentioned shortcomings of linear regression, a regularization term is introduced in Ridge regression model [12]. Ridge regression uses similar Mathematical learning model (Representation) as that of linear regression with a regularization/penalty term for the magnitude of estimated weights. The error function of linear regression is:

$$min \sum (y_{real} - y_{pred})^2 + \lambda(W)^2 \tag{3}$$

where regularization term is $\lambda(W)^2$. Which means that weight of coefficient should be as minimum as possible. If an input attribute has a small effect on improving the error function it is shut down by the penalty term.

Kernel Trick: Kernels are used to transform feature space into higher Dimensional feature space and learn a linear classifier/model in that transformed space. We use kernel functions to do this transformation implicitly.

3) Kernel Ridge Regression(KRR): Kernel Ridge Regression (KRR) differs from Ridge Regression (RR) in its form of Mathematical learning model (Representation). Input values are transformed into some higher dimensional feature space $X \rightarrow \Phi(X)$.

$$y_{pred} = W\Phi(X) + b \tag{4}$$

It is useful for problems when the features are different or larger than the number of test data points. The transformation is made computationally feasible by choice of appropriate kernels

and the "kernel trick" [13]. Most common kernels are a Radial basis function (RBF) kernel, polynomial Kernel (Poly) and linear Kernel as shown in Table 1 [12, 13]. X and X_i are two points from the dataset. In the dual space, the

Tab. 1: *some of the kernels are shown*

RBF	$(exp(-\gamma(X - X_i)^2))$
Poly	$(XX_i + b)^d$
Linear	(XX_i)

decision function is given by,

$$y_{pred} = -\frac{\alpha}{\lambda}K(X, X_i) \tag{5}$$
$$K(X, X_i) = \Phi(X) \cdot \Phi(X_i)$$

where K is the considered kernel and α is an array of Lagrange multipliers. KRR uses a similar loss function as ridge regression and kernel trick which has its origins in Support Vector Machines (SVM). That way, it can be seen as an intermediate between Ridge regression and SVM. Kernel ridge regression requires less time to train compared to SVM [14].

4) Support Vector Machines Regression (SVM Regression): In a nutshell, SVM regression differs from Kernel Ridge Regression (KRR) in its loss function which allows it to select a subset of input data set (called support vectors) which are used in the decision function. Generally, it takes longer to train the SVM model compared to KRR but results in a compact decision function.

The loss function of SVM Regression is,

$$min[1/2(W)^2 + [C\sum(\xi^* + \xi)]] \tag{6}$$

subject to:

$$[y_{real} - (W, \Phi(X)) - b] \leq \epsilon + \xi \tag{7}$$
$$[(W, \Phi(X)) + b - y_{real}] \leq \epsilon + \xi^* \tag{8}$$
$$\xi, \xi^* \geq 0 \tag{9}$$

Mathematical learning model (Representation)of SVM regression in dual space is:

$$\sum(\alpha - \alpha^*)K(X, X_i) + b \tag{10}$$

where ξ and ξ^* are slack variables to cope with constraints of the optimization. The constant $C > 0$ controls the regularization in SVR while α and α^* are Lagrange multipliers [14].

3 IPM and Machine Learning Regression

3.1 Simulation Data Details

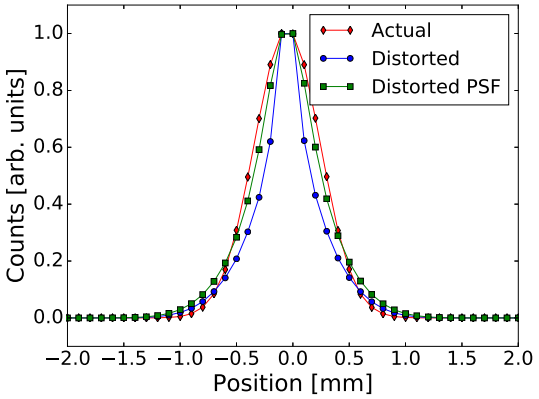


Fig. 2: *Simulation of profile distortion due to space charge using Virtual IPM.*

The Virtual-IPM simulation program has been used in order to generate training data. Table 2 shows the parameters which have been used for the simulations. σ_x, σ_y, σ_l and the bunch population N_p have been varied to cover the relevant operational region. One million

Tab. 2: *Parameters used for the simulation. σ_x, σ_y, σ_l and the bunch population have been varied within the specified intervals.*

Particle type	Protons
Energy/u	6.5 TeV
Bunch population N_p	1.1 to 1.7 $\times 10^{11}$
Bunch length σ_l (4σ)	0.9 ns to 1.2 ns
Bunch width σ_x (1σ)	0.29 mm to 0.37 mm
Bunch height σ_y (1σ)	0.4 mm to 0.6 mm
Electrode distance	85 mm
Applied voltage	4 kV
Magnetic field	0.2 T

particles were used in each simulation providing rather smooth profiles. The bin size is 0.1 mm and Gaussian point spread function with $\sigma = 0.125$ mm was used to represent the effect of optical acquisition system in the IPM system. Figure 2 shows the actual primary beam profile, the distorted profile due to space charge and the

profile read at the end of the acquisition system after application of point spread function of the optics. The input parameters to simulation were $N_p = 1.7 \times 10^{11}$, $\sigma_l = 0.9\,\text{ns}$, $\sigma_x = 0.29\,\text{mm}$, and $\sigma_y = 0.4\,\text{mm}$. There is a 20 % increase in the second central moment of the measured profile (with PSF) with respect to actual (initial) profile. Further details of virtual IPM can be found here [4].

3.2 Training

The training of ML regression algorithms is performed with three distinct parameter sources, measured profile (100 points in each profile), particle number N_p (1 point) and bunch length σ_l (1 point) as inputs forming an array of 102 points for each training sample while the output is actual width denoted by $\sigma_{x,a}$ to differentiate from predicted width $\sigma_{x,p}$. $\sigma_{y,a}$ is not used for the training process since, in any experimental usage, it will not be available as an input to the trained network for prediction of $\sigma_{x,p}$. 375 training samples were generated with a parameter scan in N_p, $\sigma_{x,a}$ and $\sigma_{y,a}$ (5 parameters each) and σ_l (3 parameters). Table 2 shows the parameter range over which training data was generated. Validation data was generated at a spacing 1%, 25% and 50% off the training data sites in each parameter space forming 372 validation samples. In addition to that 0.5% Gaussian white noise (relative to the maximum value in each parameter space) was added to the profiles to depict ADC/camera noise on the measured profile as well as measurement uncertainty on N_p and σ_l. More details can be found in [16] The training data is used to optimize the loss function and obtain the parameters of regression algorithms. sklearn library with python interface was used to define and train the ML regression algorithms discussed above. For optimization of parameters, we used grid search method by using GridSearchCV library from sklearn [14]. GridSearchCV does an exhaustive search over defined parameters with a scoring method of 'mean square error'.

3.3 Results and Discussion

For every ML Algorithm, the error% between reconstructed width and true width of the profile is calculated for the whole validation data and plotted as a histogram. All results are generated by adding Gaussian white noise=0.5% with respect to source input parameter.

Linear Regression: Figure 3 shows the histogram of error% in prediction by linear regression. where $\mu = 0.144$ and $\sigma = 0.860$

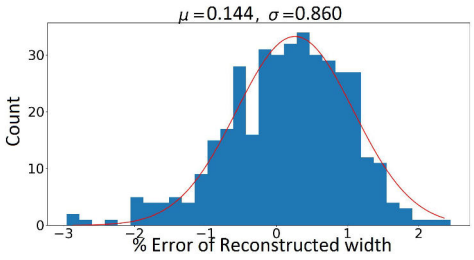


Fig. 3: *Histogram showing the percentage prediction error given by linear regression algorithm.*

Ridge Regression: In Ridge Regression after optimization using Gridsearch method [14] we obtained $\lambda = 0.0189$. Figure 4 shows the histogram of error% between reconstructed width and true width of the profile.

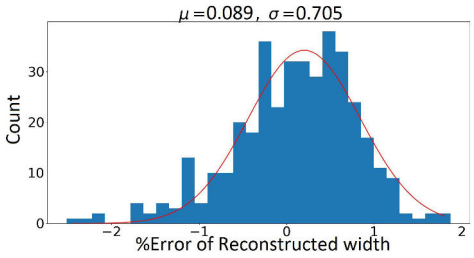


Fig. 4: *Histogram showing the percentage prediction error given by Ridge Regression Algorithm*

Kernel Ridge Regression:After training, different results are obtained with different kernels . Optimum parameters of KRR using Gridsearch were, kernel="RBF", $\lambda = 0.00106$, $\gamma = 0.1$. Figure 5 shows histogram of error% between predicted width and true width of the profile.

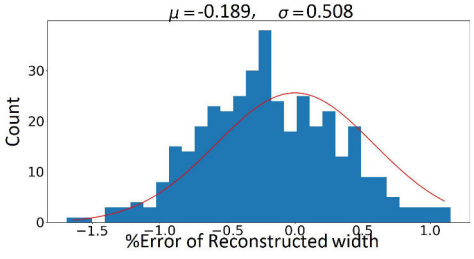


Fig. 5: *Histogram showing the percentage prediction error given by KRR Algorithm using RBF Kernel*

Support Vector Machines Regression: The optimized parameters for SVM Regression for using Gridsearch method were: kernel="RBF", $\epsilon = 0.024$, $\alpha = 0.1$, $C = 1000$. Histogram in Fig. 6 shows error% between Predicted width and true width of profile. The value of deviation

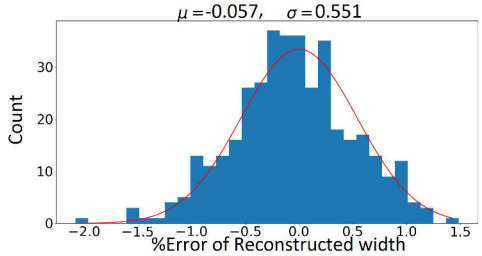


Fig. 6: *Histogram showing the percentage prediction error given by SVM using RBF Kernel*

is not significantly different for the algorithms discussed above while minor improvements with increasing complexity of the algorithm is observable.

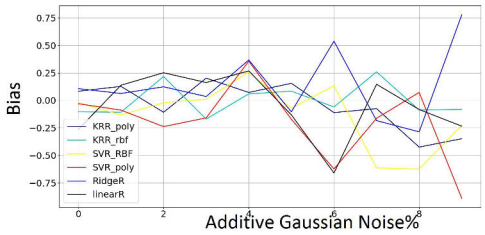


Fig. 7: *Evolution of bias of predictions with respect to noise in training and validation data*

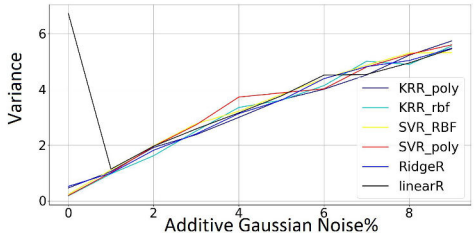


Fig. 8: *Evolution of std. deviation of predictions with respect to noise in training and validation data*

Gaussian white noise was added to both training and validation data in the range of $\sigma_{noise} = 0$ % to 10 % relative to the maximum value of each source parameter. For each set of noisy training data, training of all the ML algorithms was performed and the bias and variance of prediction error over the validation data were plotted against the added % noise. Figures 7 and 8 shows plot of bias and variance against % noise respectively. The bias of the prediction error of almost all algorithms oscillates around zero even with increased noise while the variance of prediction error is in a linear relationship with noise amplitude.

3.4 Conclusion

The reconstruction of distorted profiles for Gaussian beams was performed using machine learning regression algorithms and reconstruction errors below 1 % were obtained even with the inclusion of measurement uncertainties. Results show that compared to other machine learning regression algorithms, both SVM Regression and Kernel Ridge Regression with RBF kernel gives better prediction results. This might be due to the fact that squared exponential kernel defines a function space that is a lot larger than that of the linear kernel or the polynomial kernel.

References

[1] M. Sapinski et al., The first experience with LHC beam gas ionization monitor, Proceedings of IBIC 2012.

[2] M. Patecki et al., Electron tracking simulations in the presence of the beam and external fields, Proceedings of IPAC 2013, Shanghai, China.

[3] D. Vilsmeier et al., Investigation of the effect of beam space charge on electron trajectories in Ionization profile monitors, HB 2014.

[4] D. Vilsmeier et al., A modular application for IPM simulations, these proceedings.

[5] M. Sapinski et al., Ionization profile monitor simulations - status and future plans, Proceedings of IBIC 2016.

[6] P. Forck, Lecture notes in beam instrumentation, JUAS, 2017.

[7] K. P. Murphy, Machine Learning: A Probabilistic Perspective, The MIT Press, 2012.

[8] Sebastian Raschka,Python Machine Learning, ISBN 978-1-78355-513-0, (2015).

[9] Bishop, Christopher (2006). Pattern recognition and machine learning. Berlin: Springer. ISBN 0-387-31073-8.

[10] Ryan Tibshirani, Modern regression 1: Ridge regression Data Mining: 36-462/36-662 (2013). retrieved from: http://www.stat.cmu.edu/ ryantibs/ datamining/lectures/16-modr1.pdf

[11] Stefan Feuerriegel, Regularization Methods Business Analytics Practice Winter Term (2015/16). retrieved from: https://www.is.uni-freiburg.de/resources/business-analytics/XX_Regularization.pdf

[12] P.Paisitkriangkrai, Linear Regression and Support Vector Regression (2012). retrieved from: http://cs.adelaide.edu.au/ chhshen/ teaching/ML_SVR.pdf

[13] Justin Domke, Statistical Machine Learning Kernel Methods and SVMs, UMass (2011). retrieved from: https://people.cs.umass.edu/ domke/ courses/sml2011/07kernels.pdf

[14] Scikit-Learn Documentation 0.19.0(2017). retrieved from: http://scikit-learn.org/stable/ modules/svm.html.

[15] David A. Freedman, Statistical Models: Cambridge University Press.p.26. ISBN: 9780521112437, (2009).

[16] R. Singh et al., Proceedings of IBIC (2017).

[17] Wikipedia:Support Vector Machines Definition, retrieved from : https://en.wikipedia.org/wiki/Support_vector_machine

[18] JCGM 200:2008 International Vocabulary of Metrology Basic And General Concepts And Associated Terms (VIM)(2008). retrieved from: http://www.bipm.org/utils/common/ documents/jcgm/JCGM_200_2008.pdf

[19] Mehryar Mohri,Introduction to Machine Learning, Courant Institute and Google Research,retrieved from: http://www.cs.nyu.edu/ mohri/mlu/ mlu_lecture_1.pdf

Feedback on Student Outcomes (EAC-Computer Engineering):

Student Outcomes	Response
a) An ability to apply knowledge of Mathematics, Science, and Engineering	4
b) An ability to design and conduct experiments, as well as to analyze and interpret data	5
c) An ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability	4
d) An ability to function on multidisciplinary teams	5
e) An ability to identify, formulate, and solve engineering problems	4
f) An understanding of professional and ethical responsibility	5
g) An ability to communicate effectively	5
h) The broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context	4
i) A recognition of the need for, and an ability to engage in life-long learning	4
j) A knowledge of contemporary issues	5
k) An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice	5

Feedback on Student Outcomes (CAC-Computer Science):

Student Outcomes	Response
a) An ability to apply knowledge of Computing and Mathematics appropriate to the program's student outcomes and to the discipline	4
b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution	5
c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs	4
d) An ability to function effectively on teams to accomplish a common goal	5
e) An understanding of professional, ethical, legal, security and social issues and responsibilities	5
f) An ability to communicate effectively with a range of audiences	5
g) An ability to analyze the local and global impact of computing on individuals, organizations, and society	4
h) Recognition of the need for and an ability to engage in continuing professional development	5
i) An ability to use current techniques, skills, and tools necessary for computing practice	4
j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices	4
k) An ability to apply design and development principles in the construction of software systems of varying complexity	5