

HANDWRITTEN DIGITS RECOGNITION USING NEURAL NETWORKS

Project report submitted in partial fulfillment of the requirement
for the degree of Bachelor of Technology

in

Computer Science and Engineering/Information Technology

By

Mayank Bisht (161290)

Hritik Agarwal (161470)

SUPERVISOR- Dr.Aman Sharma

to



Department of Computer Science & Engineering and Information
Technology

Jaypee University of Information Technology

Waknaghat, Solan-173234, Himachal Pradesh

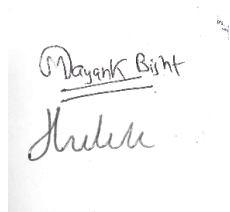
Declaration

We hereby declare that the work presented in this report entitled “ **Handwritten Digit Classification using Neural Networks**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering/Information Technology** submitted to the department of Computer Science & Engineering and Information Technology, JUIT is an authentic record of our own hardwork carried out in the timeline from January 2020 to May 2020 under the guidance of **Dr. Aman Sharma (Assistant Professor - Grade-II)**.

Content attached in the report has not been submitted for award of any other degree .It is only used for final year project report submission in JUIT.

Mayank Bisht(161290)

Hritik Agarwal(161470)

The image shows two handwritten signatures in black ink. The top signature is 'Mayank Bisht' and the bottom signature is 'Hritik Agarwal'. Both signatures are written in a cursive style.

I certify that above statement made by students is true according to my knowledge.

SUPERVISOR

Dr.Aman Sharma

The image shows a handwritten signature in black ink that reads 'Aman Sharma'.

Dated: 26 MAY 2020

ACKNOWLEDGEMENT

We want to express our deep gratitude and deep regards to our guide Dr. Aman Sharma (CSE) for his superior guidance, monitoring and constant support throughout the course of this project. The blessing, help and guidance given by him to us time to time will carry us in the journey of life on which we are about to proceed.

We are also thankful to all members of JUIT College, for the valuable information provided by them in their fields. We are grateful for their cooperation during the period of our project.

Lastly, we thank god, our parents and our classmates for their constant support without which this assignment would not have been possible.

Contents

1. Introduction

- 1.1 Introduction
- 1.2 Problem Statement
- 1.3 Objective
- 1.4 Methodology
- 1.5 Organization and Structure

2. Literature Survey

- 2.1 Books and publication
- 2.2 Talks, conferences and meetings

3. System Development

- 3.1 System Architecture
- 3.2 Perceptron
- 3.3 Prediction and loss function
- 3.4 Gradient descent and logistic regression algorithm
- 3.5 Non-Linear models
- 3.6 Neural Networks
- 3.7 Training the neural network
- 3.8 Measures to increase efficiency
- 3.9 Neural Networks using pytorch

4. Algorithm

- 4.1 Train Models
- 4.2 Process Image
- 4.3 Testing

5. Testing

5.1 Command line testing

5.2 Software testing

5.3 Dataset testing

5.4 Algorithmic testing

6. Conclusion and Future Work

List of Figures

- 1.Layers of Deep Learning[4]
- 2.Deep learning Vs Machine Learning[5]
- 3.Advantages of Deep Learning[6]
- 4.Input Image[7]
- 5.ImageNet[8]
- 6.Path for Classification[8]
- 7.Output Image[9]
- 8.5 Top Classes of Output[10]
- 9.Organization of project[11]
- 10.Layers of Application [15]
- 11.Perceptron[16]
- 12.Discrete and continuous prediction[17]
- 13.Maximum Likelihood probability[18]
- 14.Gradient descent wrt weight and bias[20]
- 15.Weight manipulation gradient[21]
- 16.Non-linear model[21]
- 17.Probability in non-linear models[22]
- 18.Weight to linear models[23]
- 19.Structure of neural networks[24]
- 20.Feed-forwarding process[25]
- 21.Procedure for back-propagation [26]
- 22.Over-fitting[27]
- 23.Early-stopping[27]
- 24.Classifier model using class[28]
- 25.Classifier model without class[29]
- 26.Feed forward and back propagation in pytorch[30]
- 27.Dropout in pytorch model[30]
- 28.Loading Image datasets[31]
- 29.Transfer Learning[32]

30.Default Transform[33]

31.Back propagation code[35]

32.Modules[36]

33.Command Line Examples[37]

34.Running python using pip[37]

List of Abbreviation

1. ANN – Artificial Neural Networks
2. CNN – Convolutional Neural Networks
3. CVPR - Computer Vision & Pattern Recognition
4. OCR - Optical Character Recognition
5. SVM – Support Vector Machine

ABSTRACT

As the world is as of now occupied with Internet for all the work and will keep on doing as such in future . Presently the unremarkable works of the users are looking for the assistance of Artificial Intelligence for its adequacy and for learning. Proceeding, AI calculation will be combined into a consistently expanding number of ordinary applications. For instance, you should need to fuse an image classifier in a Smartphone application. To do this, you would use a profound learning model arranged on incalculable pictures as an element of the general application engineering. A gigantic bit of programming improvement later on will use these sorts of models as regular pieces of employments.

In this venture, we will prepare picture classifier to arrange particular kinds of manually written digits. You can envision using something like this in a street traffic challan application that uncovers to you the number of the name plate of the vehicle your camera is investigating. You'd train this classifier, by then convey it for use in your application. We will use the dataset of manually written digits from MSIT.

CHAPTER 1: INTRODUCTION

1.1 Introduction

When it comes to recognize pictures, we people can obviously perceive and recognize distinctive features of items. This is on grounds that our brains have been trained unconsciously with a similar arrangement of pictures that has brought about advancement of capabilities to separate between things easily. We are not really conscious when we interpret real world. Experiencing distinctive elements of visual world and recognizing effortlessly is a no test to us. Our subconscious mind does every one of process with no issue.

In opposition to human brains, computer sees visuals as a variety of numerical qualities and searches for patterns in advanced picture, be it a stillvideo, realistic, or even live, to recognize and differentiate key highlights of picture. way in which a system interprets a picture is totally not quite same as humans. Computer vision utilizes picture handling calculations to analyse and understand visuals from an arrangement of pictures. Thinking about developing capability of computer vision, numerous organisation are investing into image processing to analyse and investigate information coming essentially from visual sources for various uses, for ex- medicinal image examination, identifying objects in vehicles, face detection for security reason, and so forth.

Image recognition is ability of a system to identify objects, individuals, places, and activities in images. It utilizes machine vision advances with AI and calculations to recognize images through a camera system. Much driven by ongoing progressions in ML and an expansion in computational intensity of machines, image recognition has surprised world. Car, business, retail, security observation, social insurance, cultivating and so on can have a wide utilization of picture recognition

APPLICATIONS:

Automatons:

Picture acknowledgment abilities prepared automatons can give review and vision-based programmed checking,region.

Assembling:

Exploring creation lines, evaluating fundamental concentrates constantly . Watching idea of last things to diminish defect. Assessing condition of worker may help manufacturing ventures for complete cont. of different activities in structures.

Self-governing Vehicle:

Independent vehicle having picture acknowledgment can recognize practices out on town and take significant exercises. Littler than anticipated robots can help in coordination dares to discover and trade articles beginning having 1 spot to following. It further more keeps up data of thing improvement past to shield thing from being forgotten.

Defence_Surveillance:

Recognizable proof of unforcastable activities in periphery locals and modified essential authority capacities can help check attack and realize saving lifes of officials.

These neural systems are profound neural systems that are used to orchestrate pictures , pack having comparability , and task image acknowledgment.

These systems perform optical recognitions (OCR) to digitize material and make NLP conceivable on basic and composed by hands made report, where photos are pictures being deciphered. se can be associated with when it is addressed ostensibly as a spectroogram. Convolutional systems have been associated explicitly to content assessment and diagram data with chart convolutional systems.

basic customary ML calculations is that as astounding as they may show up, in any case they 're not human like, they are machins, they need some portion of room expertise, human intercession only prepared for what y're expected for. For AI main generator and whatever remaining parts of globe, that is spot profound knowledge retained all most consummately.

DL is part of ML that satisfies unprecedented energy and flexibility by figuring out how to address globe as settled hierarchy of thoughts, with each thought described in association .

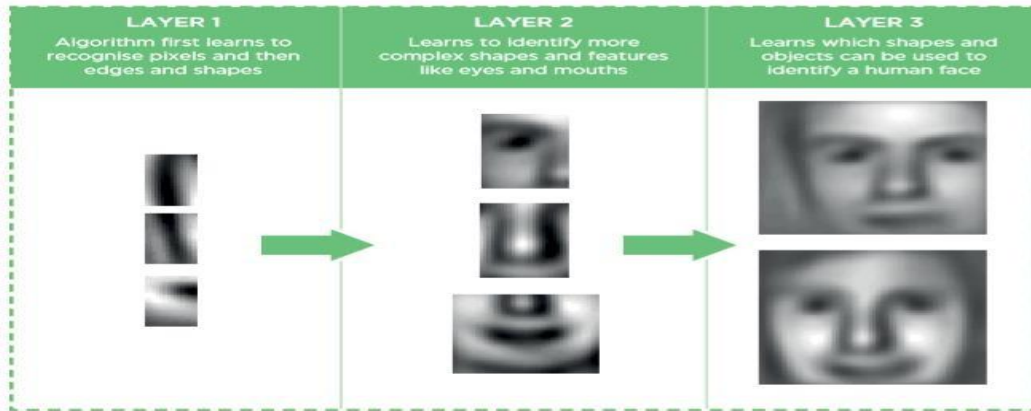


Fig. 1 - Layers of DL

1.1 Problem Statement

As prior expressed , Image acknowledgment has various applications from military to picture captcha for security and se application will keep on prospering as arrangement of pictures is turning into a need for an association. Utilization of picture modelclassifiers rar than physically grouping is practically comparative when pictures to be arranged is pitiful in number yet when we talk about ordering enormous infoset of pictures, physically characterization turns out to be exceptionally repetitive and work gets alongside incomprehensible. Likewise for physically arrangement of pictures, total information on class sets and its subtleties is obligatory generally expectation can be off base. With rise of following issues in handwritten characterization application.

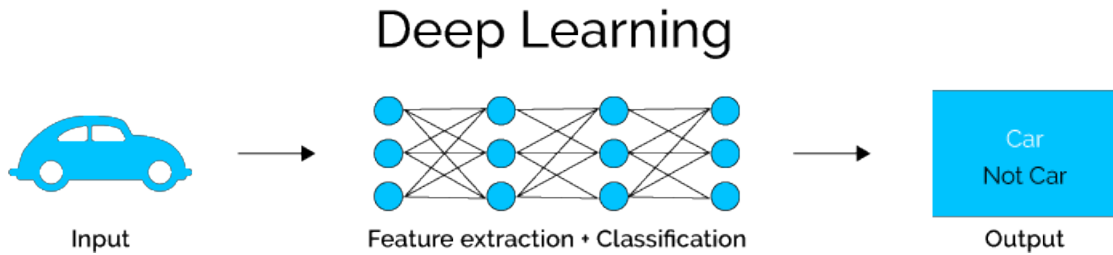
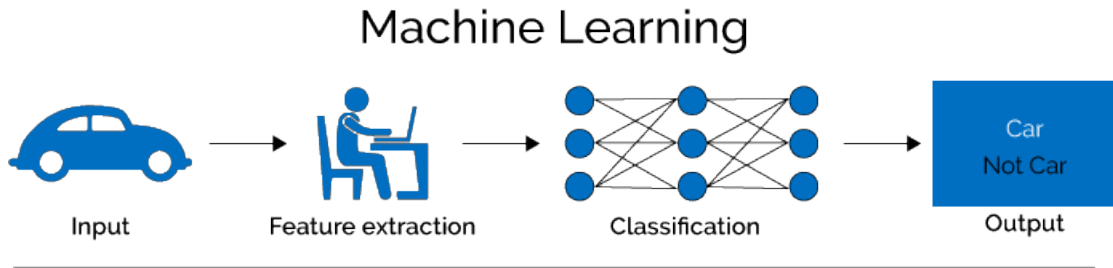


Fig two. Profound Learning versus ML

Additionally DL gives a greater exactness in outcomes as it got prepared huge number of informational indexes and give proficiency when measure of information is huge.

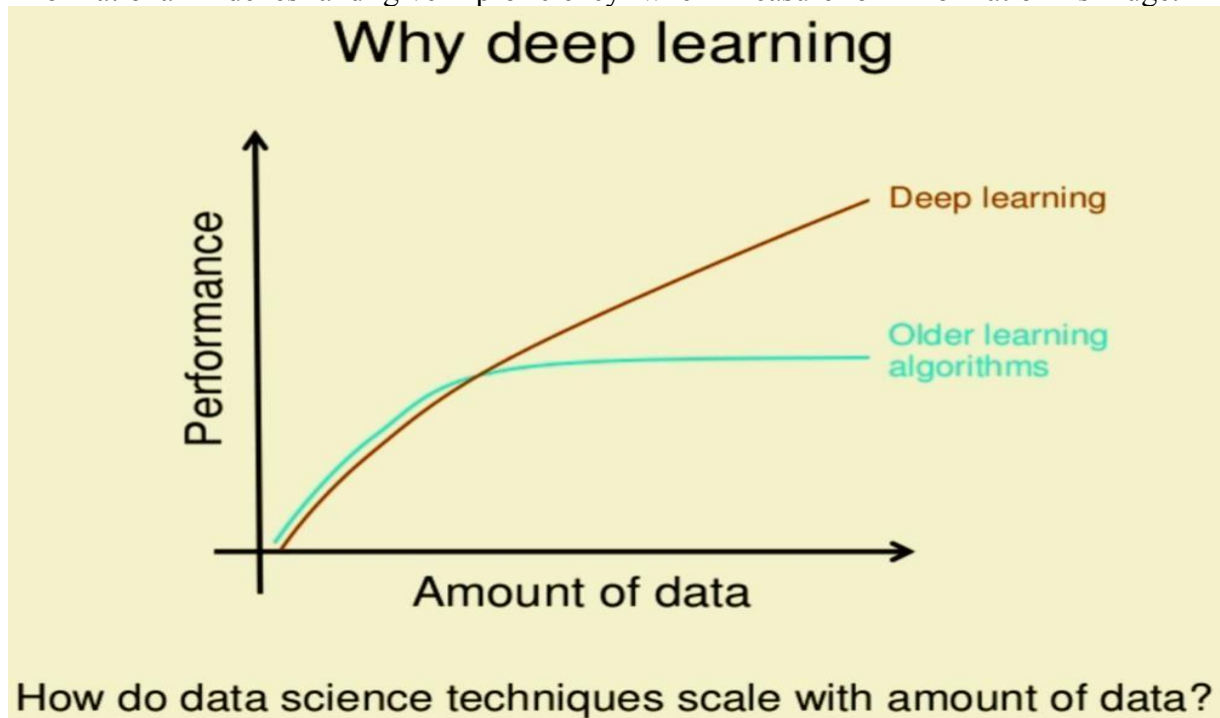


Figure3: benefits of DL

1.1 Objective

Primary target of this venture is to execute a web applications/portable utilization of a digit modelclassifier, and that modelclassifier will order pictures into ir separate names; likewise modelclassifier is prepared on a huge infoiset.

1.2 Method

As a matter of first importance, import every necessary bundles and sub bundles with reasonable epitht. Bundles necessary information in 3 sections:

- Train1: for preparing, change like scaling, revolution and trimming and so on are to be executed as it will prompt better execution. Info information ought to be resized to a specific length and width as required by pre-prepared neural network.
- Validate and test: sets can be utilized to quantify presentation of model on concealed information.

Image Recognition Roadmap

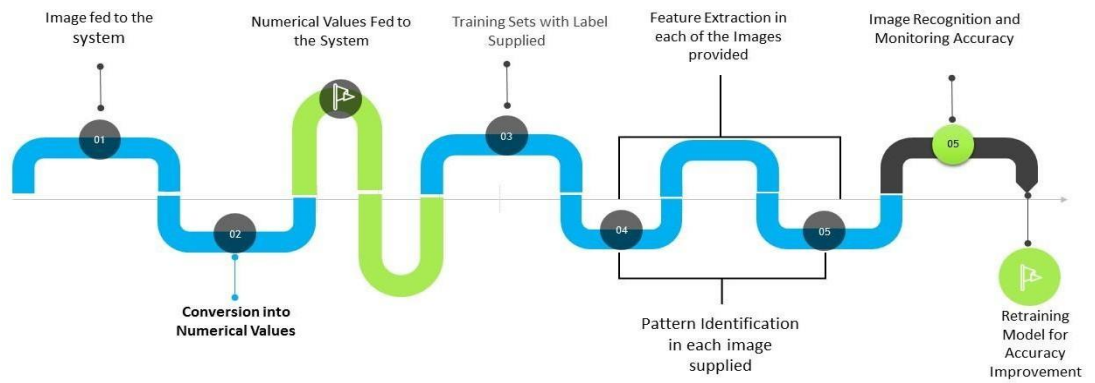


Fig. 6 – classification path

Preparing and building modelclassifier:

We will utilize pre-prepared model from torchvision.models to get highlights of picture. (straightforward portrayal VGG can be utilized)

- Load a pre-prepared system.
- Make anor, untrained system as a modelclassifier
- use backpropagation to prepare modelclassifier layers utilizing pre-prepared system to get highlights.
- track precision and misfortune to decide best hyperparameters and look after insights.

Forecast of classes:

Presently, probabilities of diverse class is determined to discover class of picture. Number ofclasses can utilize k-top capacity.

Where k is a number, 5-top characterizes top 5 classes and 4-top characterizes 4 top classes which is like picture.

1.1 Organization

venture is separated into numerous means:

1. process and upload picture info set
2. On info set, train model classifier
3. forecast picture content utilizing model classifier prepared in two.
4. Analyse outcomes gave by design

CHAPTER TWO: LITERARY SURVEY

2.1 Books and Publication

To land at an end to utilize specific sort of order method in undertaking required a hard investigation of all likely procedures with ir favorable circumstances and disservices alongside ir inspiration.

As to arrange any picture different advances is included:

- Define Classification Classes

Arrangement classes relies upon property and goal of picture.

- Feature Selection

Multi-orworldly and multi-worldly properties is utilized to address contrasts between classes. Highlights of classes fluctuates.

- Train1 information' test

It is important to test preparation information. Administered grouping, Unsupervised learning, and Semi directed learning will be utilized by information.

- Estimate Universal Statistics

Proper technique will be chosen with assistance of look into.

- Method of Classification

A few techniques are-Linear, SVM,decision tree ,naïve bayes, Fuzzy Tree.

Additionally, IEEE paper likewise secured this me " we audit present movement of picture arrangement strategies and systems. Picture order is an exceptionally intricate procedure which relies on different variables. Here, we examine about present procedures, issues just as possibilities of picture order. principle spotlight will be on cutting edge arrangement systems which are utilized for improving order exactness just as unwavering quality."

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 System Structure (Perceptron)

It resembles neuron of body is structure square of AI. It takes inputs (quantities of sources of info are not restricted) and data sources are duplicated by certain load and then blends of information sources and load is then prepared by it .



Example-Let a model says that a person will get admitted in college based on test and grade score.

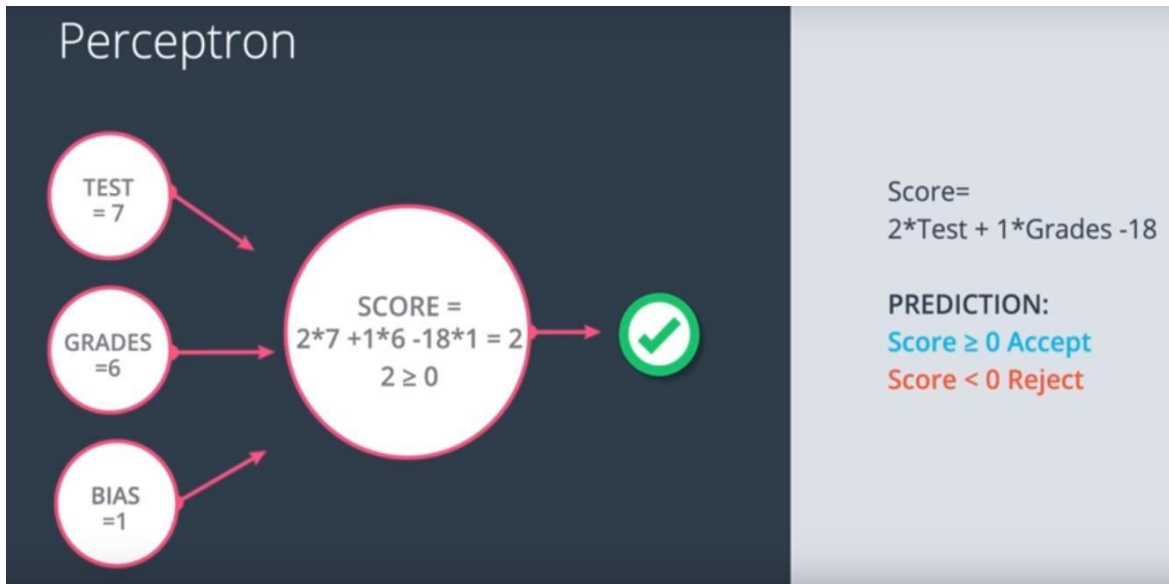


Fig 11:Perceptron

In this model, inputs are test and grade of applicant, and information sources are nourished to it and as per past information a direct condition is shaped by it which will pass judgment on score of competitor. In this figure score equivalent to or above

0 methods Acceptance from college and score beneath 0 methods Denial from college.

As should be obvious straight condition will choose result of this.

Along lines, is particularly risks that straight condition may foresee yield wrong. To maintain a strategic distance from this off-base yield condition should be changed with assistance of uploads.

re are two kinds of falseforecast:-

1. If fact of matter is grouped positive, yet it has a negative name.
2. If fact of matter is grouped negative, yet it has a positive mark. What's more, these two sorts of mistake has distinctive outcome set.

PERCEPTRON PROCEDURE:

For each notclassified point(y_1, y_2, \dots, y_m):

1: If $\text{pred} == 0$:

for $j=1$ to m :

alter $k(j)$ to $k(j) + ay(j)$

alter $b \rightarrow b + a$

two: If $\text{pred} == 1$:

for $j=1$ to m :

alter $k(j) \rightarrow k(j) - ay(j)$ and alter

$b \rightarrow b - a$

Forecast and loss function :

we will utilize a sigmoid capacity which gives us a constant worth extending somewhere in range of 0 and 1.

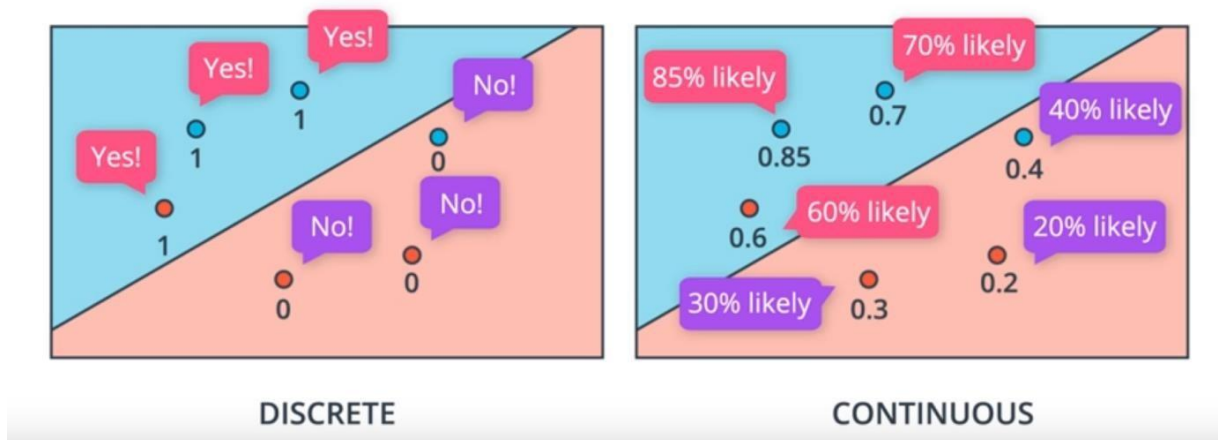


Figure12: Discrete and contiguous forecast

focuses on hold has estimation of 0.5 and as focuses move away from line n incentive for focuses fluctuate as point increments as fact of matter is over line and diminishes as fact of matter is beneath line.

For eg. In figure over, those focuses over line has likelihood 0.60 , 0.70 and 0.8500. Till now, we have just two yield class yet imagine a scenario where we have to characterize focuses into at least three classes. Think????u got that ->We will utilize softmax1 work.

Softmax1 function for A class:

e^a

$$\frac{e^a}{e^a + e^b + e^c}$$

Softmax1 function for B class: $\frac{e^b}{e^a + e^b + e^c}$ Softmax1 function for C class:

$$\frac{e^c}{e^a + e^b + e^c}$$

Greatest alike Probab-This is result of individual probabilities their particular shading as determined by perceptron. for ex-:

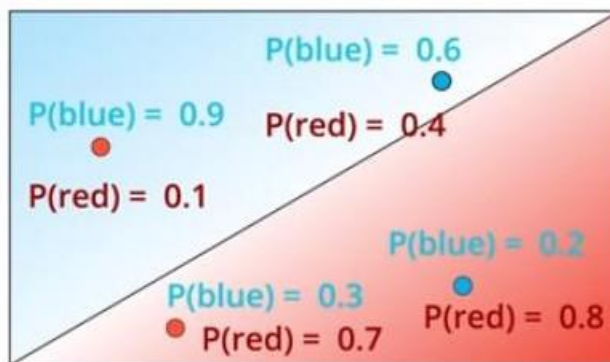


Fig 13: Maximum Likelihood Probability

$$MLP1 = \{(0.1) * (0.6) * (0.7) * (0.2)\} = 0.0084$$

In event that quantity of focuses are excessively enormous, at that point result of likelihood can be little and will have no essentialness, to maintain a strategic distance from this circumstance we have to whole individual probabilities, this offered ascend to idea of cross entropy.

In CROSS ENTROPY, we take log of all likelihood and include m. In model above, cross entropy can be determined by:

$$\text{Log} = \ln \{[-\log(0.6) - \log(0.2) - \log(0.1) - \log(0.7)]\} = 4.8$$

(To summarize):

Now, we can go ahead and calculate the derivative of the error E at a point x , with respect to the weight w_j .

$$\begin{aligned}
 \frac{\partial}{\partial w_j} E &= \frac{\partial}{\partial w_j} [-y \log(\hat{y}) - (1 - y) \log(1 - \hat{y})] \\
 &= -y \frac{\partial}{\partial w_j} \log(\hat{y}) - (1 - y) \frac{\partial}{\partial w_j} \log(1 - \hat{y}) \\
 &= -y \cdot \frac{1}{\hat{y}} \cdot \frac{\partial}{\partial w_j} \hat{y} - (1 - y) \cdot \frac{1}{1 - \hat{y}} \cdot \frac{\partial}{\partial w_j} (1 - \hat{y}) \\
 &= -y \cdot \frac{1}{\hat{y}} \cdot \hat{y}(1 - \hat{y})x_j - (1 - y) \cdot \frac{1}{1 - \hat{y}} \cdot (-1)\hat{y}(1 - \hat{y})x_j \\
 &= -y(1 - \hat{y}) \cdot x_j + (1 - y)\hat{y} \cdot x_j \\
 &= -(y - \hat{y})x_j
 \end{aligned}$$

A similar calculation will show us that

$$\frac{\partial}{\partial b} E = -(y - \hat{y})$$

Fig 14: GD regarding weight error and bias error

Gradient Descent Step

Therefore, since the gradient descent step simply consists in subtracting a multiple of the gradient of the error function at every point, then this updates the weights in the following way:

$$w'_i \leftarrow w_i - \alpha[-(y - \hat{y})x_i],$$

which is equivalent to

$$w'_i \leftarrow w_i + \alpha(y - \hat{y})x_i.$$

Similarly, it updates the bias in the following way:

$$b' \leftarrow b + \alpha(y - \hat{y}),$$

Fig 15: weight manipulated regarding grad.

3.3 Non-Linear Models

Till now we have just discussed remembering a straight model however Neural Network are utilized to achieve a progressively mind boggling organized errand and se assignments are for most part non-direct. To accomplish a non-straight model ,incidentally ,a direct model is being utilized

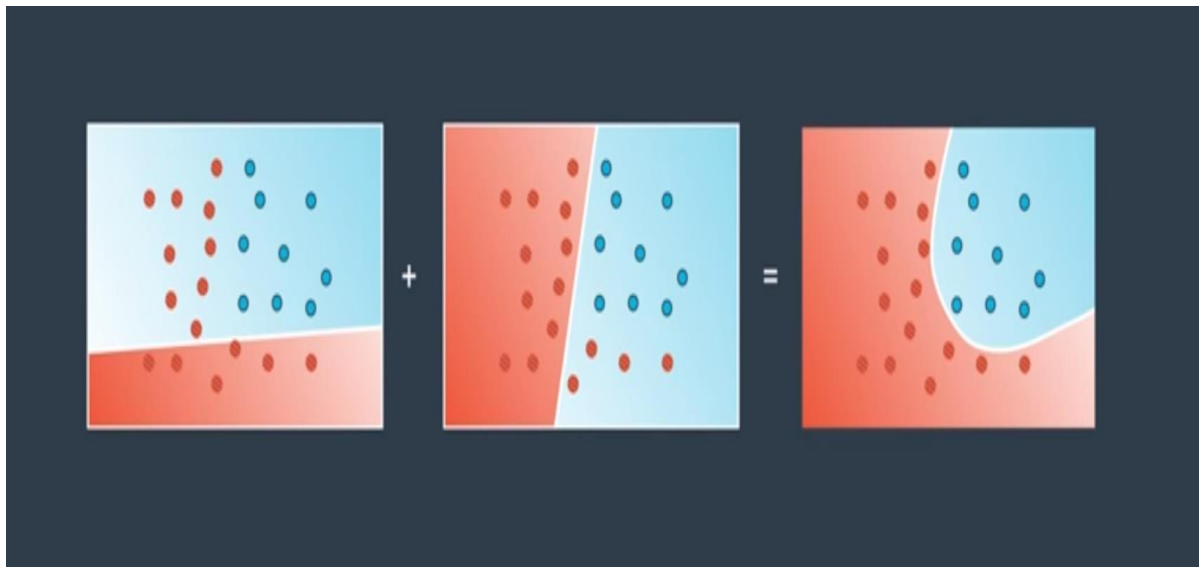


Fig 16: Non-Linear models

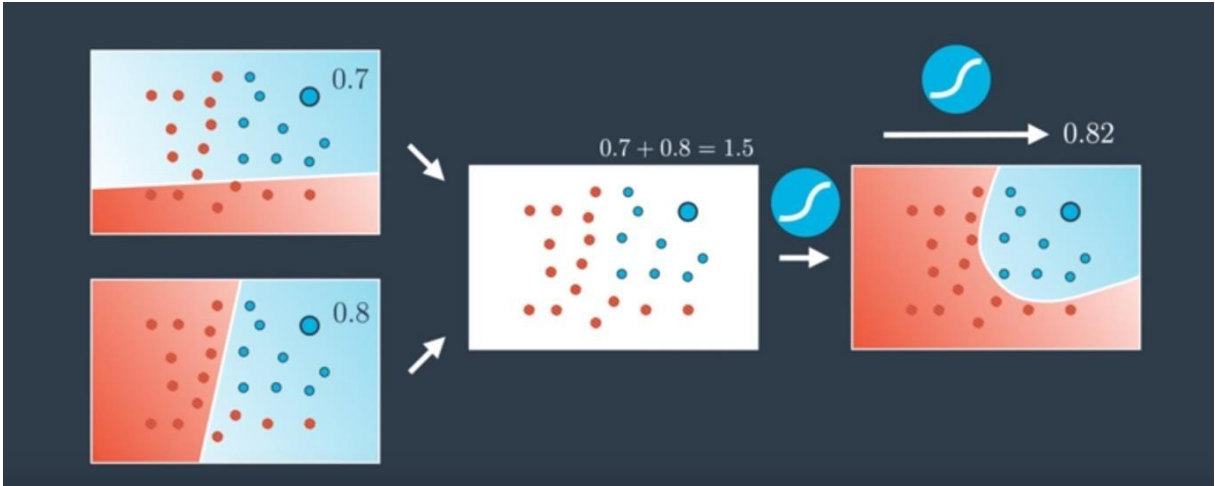


Fig 17: Probab in non-linear machine models

We can likewise add uploads and inclination to models like in above graph we can add a upload of 7 to initially demonstrate and a upload of 5 to second show and yield is determined as needs be.

$$\{ \{ [7*(0.7)+5*(0.8)- 6]=two.9 ->Sigmoid-> 0.95 \} \}$$

can characterize ideas of non-direct model and uploads to straight figure shown as:

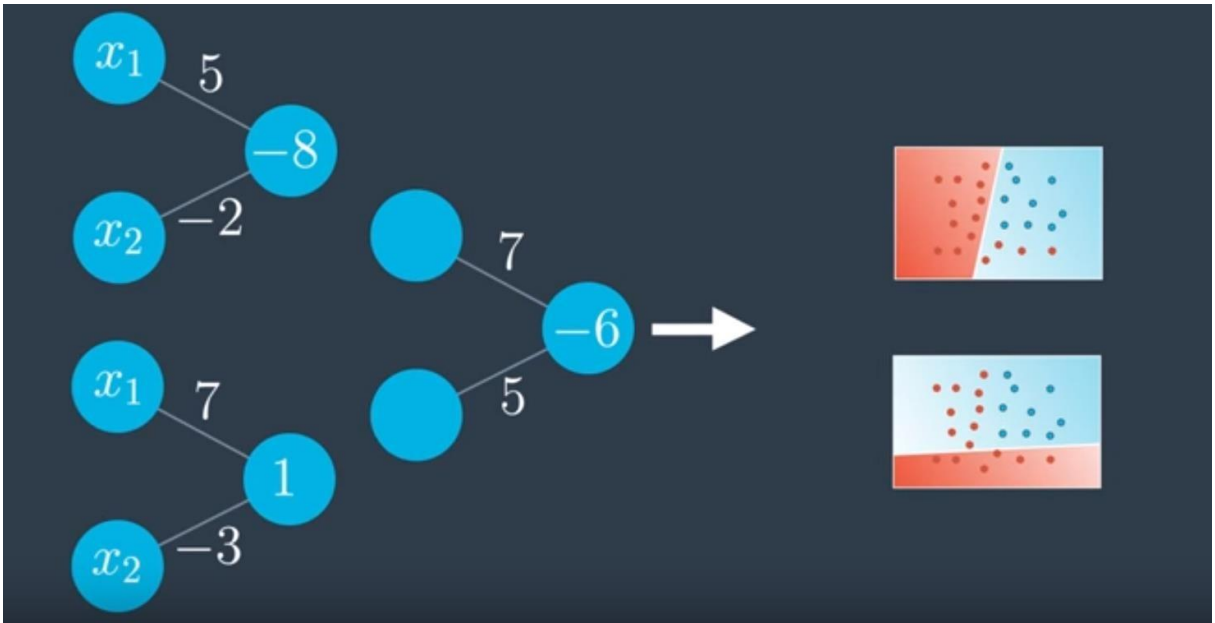


Fig 18: uploads to linear machine models

Every one of these ideas are essentially significant and base ideas of Deep Neural Network.

3.4 Neural Network:

Utilizing every one of se ideas of perceptrons and non-direct models we can characterize a neural system. Neural Network involves 3 Layers:

1. 1INPUT
2. 2HIDDEN
3. 3OUTPUT

Information layer contains information gave by model which thus are increased by uploads and this amalgam turns into contribution for Hidden layers. re can be many shrouded layers of various shapes. se shrouded layers are utilized to shape extremely complex models. yield of concealed layers turns into contribution for yield layer. quantity of yields in yield layers is by and large equivalents to classes of info information.

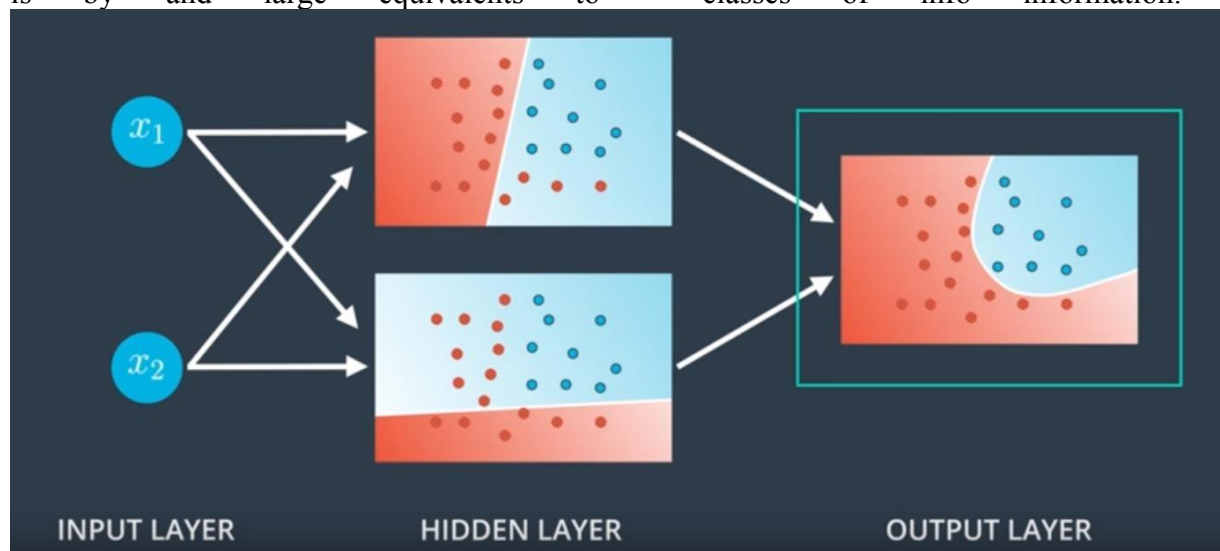


Fig 19: Neural Network(structure)

3.5 Train1 Neural Network:

A neural system should be prepared before effectively arranging pictures. Preparing a Neural systems commonly comprise of two stages:

1. Feed-Forwarding
2. Backpropagation

1. Feed-Forwarding- explanation se systems are called feedforward is that progression of data happens forward way, as x is utilized to figure some halfway capacity in shrouded layer which thusly is utilized to compute y.

se systems are spoken to by an organization of a wide range of capacities. Each model is related with a diagram portraying how capacities are created toger.

Fundamentally, Feedforward is procedure neural systems use to transform contribution to a yield. Sources of info are first duplicated with ir separate uploads and afterward enactment work is applied to se spot item. yield of enactment work turns into contribution for shrouded layer and afterward se info are again spot produced with ir separate uploads and subsequently yield becomes contribution for concealed layer assuming any and on off chance that not, at that point se yield goes directly to yield layer.

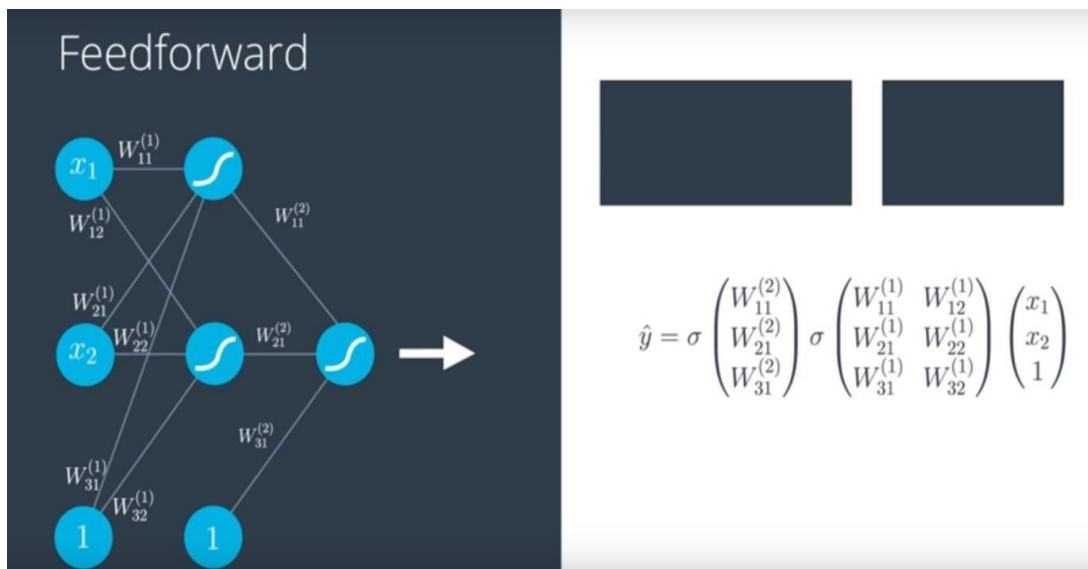


Figure20: Feed_forward procedure

2. Backpropagation - Backpropagation calculations are strategies used to prepare neural systems following an angle plunge approach that uses chain rule. primary element of backpropagation is its iterative, recursive and effective method for ascertaining weight refreshes which improves system until it can play out errand for which it is being made.

In backpropagation, mistakes are determined by looking at yield of feed-sending and names. Inclination of blunder concerning uploads are n determined. Angle regarding weight is itself a chain subsidiary as mistake is associated with yield , yield is associated with initiation capacity and enactment work is associated with uploads. This slope is utilized to limit blunder by refreshing uploads of system. This procedure of feed sending and backpropagation is run various occasions. To dodge any uncommon weight changes , we use learning rate speed with goal that weight don't change on extraordinary level and uprightness just as effectiveness of model ought to be kept up.

For now we'll only consider a simple network with one hidden layer and one output unit. Here's the general algorithm for updating the weights with backpropagation:

- Set the weight steps for each layer to zero
 - The input to hidden weights $\Delta w_{ij} = 0$
 - The hidden to output weights $\Delta W_j = 0$
- For each record in the training data:
 - Make a forward pass through the network, calculating the output \hat{y}
 - Calculate the error gradient in the output unit, $\delta^o = (y - \hat{y})f'(z)$ where $z = \sum_j W_j a_j$, the input to the output unit.
 - Propagate the errors to the hidden layer $\delta_j^h = \delta^o W_j f'(h_j)$
 - Update the weight steps:
 - $\Delta W_j = \Delta W_j + \delta^o a_j$
 - $\Delta w_{ij} = \Delta w_{ij} + \delta_j^h a_i$
- Update the weights, where η is the learning rate and m is the number of records:
 - $W_j = W_j + \eta \Delta W_j / m$
 - $w_{ij} = w_{ij} + \eta \Delta w_{ij} / m$
- Repeat for e epochs.

Figure21: procedure for back_propagation

3.6 Measures to build efficient Model:

Early halting: When we train our information countless occasions, misfortune starts to diminish however when we attempt to run approve information on prepared model, misfortune diminishes up somewhat and afterward it starts to increment. This is on grounds that we have prepared information with countless ages and model has become an overfitted model which won't perform proficiently on approval information.

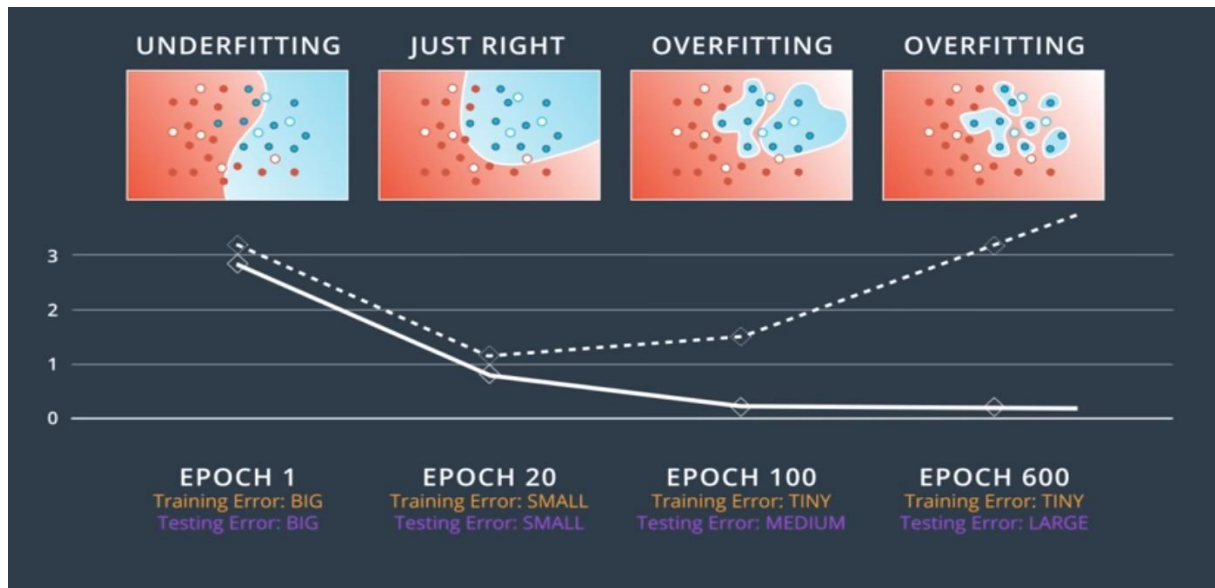


Figure :Over_fitting

To maintain a strategic distance from over fitting we have to stop our ages tally where misfortune for substantial information begins expanding. This is called early halting procedure.

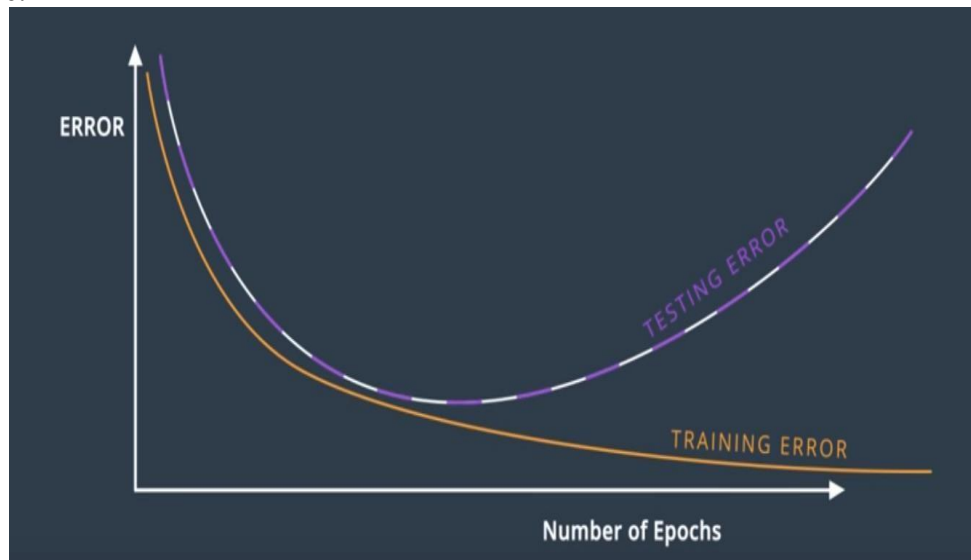


Fig :Early_halting

(_Dropout):In dropout, arbitrary hubs in each epochs¹ are solidified and not permitted to prepare with goal that different hubs take larger part all while and gets solid, way toward choosing hubs for solidifying is totally irregular and is finished by altering a likelihood like .two,.3,etc.

Minima(local): While computing angle plunge, you will consistently not have option to set a worldwide minima, you may experience a nearby minima. To abstain from getting a local_minima, you can restart getting slope drop from irregular places with goal that you won't get issue.

3.7 Neural Network utilizing Pytorch_

Building system: Using pytorch, we can structure our own system which involves quantity of sources of info, concealed layers, number of shrouded layers, yield layers, and actuation capacities.

```
class Network(nn.Module):
    def __init__(self):
        super().__init__()
        # Defining the layers, 128, 64, 10 units each
        self.fc1 = nn.Linear(784, 128)
        self.fc2 = nn.Linear(128, 64)
        # Output layer, 10 units - one for each digit
        self.fc3 = nn.Linear(64, 10)

    def forward(self, x):
        ''' Forward pass through the network, returns the output logits '''

        x = self.fc1(x)
        x = F.relu(x)
        x = self.fc2(x)
        x = F.relu(x)
        x = self.fc3(x)
        x = F.softmax(x, dim=1)

        return x

model = Network()
model
```

Figure24_: Modelclassifier machine model utilizing_class

Here, we have characterized a class Network which takes nn.Module as method argument. Here we have characterized two shrouded or we can say hidden layers with inputs 128 and 64 separately. Also, after each speck result of uploads and sources of info, initiation work is performed. yield comprise of 10 classes and can be acquired utilizing Softmax.

A system can likewise be made utilizing `nn.Sequential` in which we don't need to characterize a class.

```
model = nn.Sequential(nn.Linear(784, 128),
                      nn.ReLU(),
                      nn.Linear(128, 64),
                      nn.ReLU(),
                      nn.Linear(64, 10))
```

Figure25: Modelclassifier machine model without_class

Pytorch losses:

We can ascertain misfortune with PyTorch utilizing `nn` module. PyTorch gives misfortunes to model cross-entropy misfortune (`nn.CrossEntropyLoss`). In general, misfortune is appointed to basis. To really ascertain misfortune, you initially characterize model at that point go in yield of your system and right labels. yield of measure is misfortune for system.

Auto_gradand losses simultaneously:

```
epochs = 5
for e in range(epochs):
    running_loss = 0
    for images, labels in trainloader:
        images = images.view(images.shape[0], -1)
        optimizer.zero_grad()
        output = model.forward(images)
        loss = criterion(output, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainloader)}")
```

Figure26: feed_forward and back_propogation (pytorch)

Here criteria figures loss and `loss.backward()` computes angle_descent and `optimizer.step()` refreshes uploads of ML model.

Derivation and approval:

To expand effectiveness of Pytorch model we can utilize `nn.Dropout` work which enables hubs to adapt all more precisely.

```
class Classifier(nn.Module):
    def __init__(self):
        super().__init__()
        self.fc1 = nn.Linear(784, 256)
        self.fc2 = nn.Linear(256, 128)
        self.fc3 = nn.Linear(128, 64)
        self.fc4 = nn.Linear(64, 10)

        # Dropout module with 0.2 drop probability
        self.dropout = nn.Dropout(p=0.2)
```

Figure27_: Drop-out(pytorch)

For approval, we check model with concealed pictures and with angle off in light of fact that objective here is to discover precision of model and not preparing. `model.eval()` coordinates model for assessment reason.

Pictures in model are in same path as of preparing and figure approval loss, we will ascertain likelihood for each yield class and discover class with most extreme likelihood and contrast it and mark. Along se lines we can compute exactness and approval misfortune precisely.

Stacking Image_Infosets:

```

data_dir = 'Cat_Dog_data'

train_transforms = transforms.Compose([transforms.RandomRotation(30),
                                       transforms.RandomResizedCrop(224),
                                       transforms.RandomHorizontalFlip(),
                                       transforms.ToTensor()])

test_transforms = transforms.Compose([transforms.Resize(255),
                                       transforms.CenterCrop(224),
                                       transforms.ToTensor()])

train_data = datasets.ImageFolder(data_dir + '/train', transform=train_transforms)
test_data = datasets.ImageFolder(data_dir + '/test', transform=test_transforms)

trainloader = torch.utils.data.DataLoader(train_data, batch_size=32)
testloader = torch.utils.data.DataLoader(test_data, batch_size=32)

```

Figure28: StackingImage_infoSETS(uploading)

Here, we have stacked train and test information independently utilizing the ImageFolder module of torchvision. We have utilized the ImageFolder strategy to get the registry. In wake of stacking information we have to drive information in the DataLoader strategy for torch.utils.data with a contention of group size.

Transfer_learning:

ImageNet is a huge dataset with more than 1 million named pictures in 1000 classes. It is utilized to prepare profound neural systems utilizing an engineering called convolutional layers. When prepared, these models work amazingly well as highlight locators for pictures they weren't prepared on. Utilizing a pre-prepared system on pictures not in the preparation set is called transfer learning.

We can import pretrained models utilizing torchvision.models and in contention we need to give pretrained=True. At the point when we upload a pre-prepared model we need to set its parameters. In any case, stacking a pretrained model doesn't take care of our concern. Still we need to characterize our parameters like information, yield and concealed layers.

Section 4: ALGORITHMS

Algo1: **Train1 model**

Characterize train1_model(infoset,structure,hyperparameters)

->Define Datauploaders utilizing imagesets

->Calculate infoset sizes

->Upload machine_model

->device= GPU (if accessible)

->else Use-CPU

->start_time

->train1

->stop_time

->statistic

Likewise on off chance that contentions are given through direction line, at that point

estimation of data_transforms are predefined.

```
# Train model if invoked from command line
if args.data_dir:
    # Default transforms for the training, validation, and testing sets
    data_transforms = {
        'train': transforms.Compose([
            transforms.RandomRotation(45),
            transforms.RandomResizedCrop(224),
            transforms.RandomHorizontalFlip(),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ]),
        'valid': transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ]),
        'test': transforms.Compose([
            transforms.Resize(256),
            transforms.CenterCrop(224),
            transforms.ToTensor(),
            transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
        ])
    }
```

Figure30 - Default change

Algo two: Process_Image

This algo is utilized to change and process pictures with goal that every one of pictures have same detail.

def image_process(image):

- >open picture
- >resize picture
- >center_crop picture
- >to_tensor

Algo3: Back_propogation

```
1 import numpy as np
2 from data_prep import features, targets, features_test, targets_test
3
4 np.random.seed(21)
5
6 def sigmoid(x):
7     """
8     Calculate sigmoid
9     """
10    return 1 / (1 + np.exp(-x))
11
12
13
14 n_hidden = 2 # number of hidden units
15 epochs = 900
16 learnrate = 0.005
17
18 n_records, n_features = features.shape
19 last_loss = None
20 # Initialize weights
21 weights_input_hidden = np.random.normal(scale=1 / n_features ** .5,
22                                         size=(n_features, n_hidden))
23 weights_hidden_output = np.random.normal(scale=1 / n_features ** .5,
24                                         size=n_hidden)
25
26 for e in range(epochs):
27     del_w_input_hidden = np.zeros(weights_input_hidden.shape)
28     del_w_hidden_output = np.zeros(weights_hidden_output.shape)
29     for x, y in zip(features.values, targets):
30         ## Forward pass ##
31
32         hidden_input = np.dot(x, weights_input_hidden)
33         hidden_output = sigmoid(hidden_input)
34         output = sigmoid(np.dot(hidden_output, weights_hidden_output))
35
36         ## Backward pass ##
37
38         error = y - output
39         output_error_term = error * output * (1 - output)
40
41         ## propagate errors to hidden layer
42
43
44         hidden_error = np.dot(weights_hidden_output, output_error_term)
45         hidden_error_term = hidden_error * hidden_output * (1 - hidden_output)
46
47
48         del_w_hidden_output += output_error_term * hidden_output
49         del_w_input_hidden += hidden_error_term * x[:, None]
50
51
52 weights_input_hidden += learnrate * del_w_input_hidden / n_records
53 weights_hidden_output += learnrate * del_w_hidden_output / n_records
54
55 # Printing out the mean square error on the training set
56 if e % (epochs / 10) == 0:
57     hidden_output = sigmoid(np.dot(x, weights_input_hidden))
58     out = sigmoid(np.dot(hidden_output,
59                          weights_hidden_output))
60     loss = np.mean((out - targets) ** 2)
61
62     if last_loss and last_loss < loss:
```

```

63         print("Train loss: ", loss, " WARNING - Loss Increasing")
64     else:
65         print("Train loss: ", loss)
66         last_loss = loss
67
68     # Calculate accuracy on test data
69     hidden = sigmoid(np.dot(features_test, weights_input_hidden))
70     out = sigmoid(np.dot(hidden, weights_hidden_output))
71     predictions = out > 0.5
72     accuracy = np.mean(predictions == targets_test)
73     print("Prediction accuracy: {:.3f}".format(accuracy))

```

Figure31:back_propagation (code)

Part 5: TEST

First thing to recollect is to import each library which we have utilized in python application, with goal that pointless blunder of capacity not discovered is hindering aggregating procedure.

Prior to aggregating, make a point to import each library in record:

```
import matplotlib.pyplot as plt
import numpy as np
import torch
from torch import nn
from torch import optim
import torch.nn.functional as F
from torchvision import datasets, transforms, models
import torchvision.models as models
from PIL import Image
import json
from matplotlib.ticker import FormatStrFormatter
```

Figure32 - Module

5.1 Command_line test

Order(command) line not only supportive in accumulating and executing record however it can likewise be utilized in passing parameters

- > Directory can be passed in order line alongside document name
- >structure type can be passed in direction line
- > set hyperparameter
- >GPU determination
- >k-no doubt case
- >map to genuine name

- Train a new network on a data set with `train.py`
 - Basic Usage : `python train.py data_directory`
 - Prints out current epoch, training loss, validation loss, and validation accuracy as the network trains
 - Options:
 - Set directory to save checkpoints: `python train.py data_dir --save_dir save_directory`
 - Choose architecture (alexnet, densenet121 or vgg16 available): `python train.py data_dir --arch "vgg16"`
 - Set hyperparameters: `python train.py data_dir --learning_rate 0.001 --hidden_layer1 120 --epochs 20`
 - Use GPU for training: `python train.py data_dir --gpu gpu`
- Predict flower name from an image with `predict.py` along with the probability of that name. That is you'll pass in a single image `/path/to/image` and return the flower name and class probability
 - Basic usage: `python predict.py /path/to/image checkpoint`
 - Options:
 - Return top K most likely classes: `python predict.py input checkpoint --top_k 3`
 - Use a mapping of categories to real names: `python predict.py input checkpoint --category_names cat_to_name.json`
 - Use GPU for inference: `python predict.py input checkpoint --gpu`

Figure33 –Command_line models

5.2 Software_Test

Tools like VGG, Resnet ,Alexnet are downloaded through pytorch.

```

To install pip run in the command Line

python -m ensurepip -- default-pip

to upgrade it

python -m pip install -- upgrade pip setuptools wheel

to upgrade Python

pip install python -- upgrade

Additional Packages that are required are: Numpy, Pandas, Matplotlib, Pytorch, PIL and json.
You can download them using pip

pip install numpy pandas matplotlib pil

or conda

conda install numpy pandas matplotlib pil

```

Fig34- run python utilizing pip

To utilize program on PC needs Anaconda, Gitbash and python introduced. Gitbash will go about as Terminal like in Ubuntu and Mac.

An outsider Code proofreader is required for execution of program.

5.3 Data-Set test

Extent of preparing and test ought to be adequate for both area to make productive program.

A info set is to be separated into 3 sections:

1-Test

2-Train1

3-Validate

Preparing info set is utilized to prepare model, parcels and bunches of information is utilized to prepare model, increasingly insightful and exact will be model. Be that as it may, an equalization ought to be between train1 information and test information to prepare model viably and to test for high precision.

By and large 80-two0 is most adequately utilized extent with 80% of all out information being utilized for preparing and rest two0% for test reason.

5.4 Algorithmic_Test

How to forecast class of picture is top two classes in outcome is of equivalent likelihood?

Calculation ought to be intended to characterize class of a sudden death round picture by contrasting picture and both classes' pictures and whichever class get higher precision is class of tie_breaker images.

Part 6: CONCLUSION AND FUTURE WORKS

Conclusion: In task, we have executed various kinds of systems when it is self-made or pre-prepared systems and in those systems, exactness differs over enormous range for instance in independent system precision regularly goes from 0.7 to 0.8 and in independent system with controlled epoch, approval information, and dropout modules exactness normally runs from 0.8 to 0.85. To additionally expand this, a trained arrangement has been executed which has been prepared over PictureNet information, and precision has been found in scope of above 90%. To close, as far as grouping of pictures, if model is little i.e. no. of info and yield is less, than self-planned order can be actualized yet in event that system model is enormous, for example, number of yield is exceptionally huge in number than it is smarter to utilize trained set.

Future works: Although a great deal of work should be possible as far as expanding exactness of self-made machine model, however explicitly a lot of work is possible in accuracy gain.

Screenshot:

```
localhost:8888/notebooks/udacity/trainmnist.ipynb

jupyter trainmnist Last Checkpoint: Last Monday at 9:00 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [1]: import torch
        from torch import nn, optim
        import torch.nn.functional as F
        from torchvision import datasets, transforms

        # Define a transform to normalize the data
        transform = transforms.Compose([transforms.ToTensor(),
                                      transforms.Normalize([0.5], [0.5])
                                      ])

        # Download and load the training data
        trainset = datasets.MNIST('~/.pytorch/MNIST_data/', download=True, train=True, transform=transform)
        trainloader = torch.utils.data.DataLoader(trainset, batch_size=64, shuffle=True)

In [3]: model = nn.Sequential(nn.Linear(784, 128),
                              nn.ReLU(),
                              nn.Linear(128, 64),
                              nn.ReLU(),
                              nn.Linear(64, 10),
                              nn.LogSoftmax(dim=1))

        criterion = nn.NLLLoss()
        optimizer = optim.SGD(model.parameters(), lr=0.003)

        epochs = 5
        for e in range(epochs):
            running_loss = 0
            for images, labels in trainloader:

                images = images.view(images.shape[0], -1)
                optimizer.zero_grad()

                output = model.forward(images)
                loss = criterion(output, labels)
                loss.backward()
                optimizer.step()

                running_loss += loss.item()
```

```
jupyter trainmnist Last Checkpoint: Last Monday at 9:00 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

optimizer.zero_grad()

        output = model.forward(images)
        loss = criterion(output, labels)
        loss.backward()
        optimizer.step()

        running_loss += loss.item()
    else:
        print(f"Training loss: {running_loss/len(trainloader)}")

Training loss: 1.8599593959637541
Training loss: 0.7988998135333376
Training loss: 0.5108988250433001
Training loss: 0.4217396202340309
Training loss: 0.37978794981739417

In [4]: %matplotlib inline
        import helper

        images, labels = next(iter(trainloader))

        img = images[0].view(1, 784)
        # Turn off gradients to speed up this part
        with torch.no_grad():
            logits = model.forward(img)

        # Output of the network are logits, need to take softmax for probabilities
        ps = torch.exp(logits)
        helper.view_classify(img.view(1, 28, 28), ps)
        print(labels[0])
        print(ps)

tensor(2)
tensor([[4.2455e-05, 1.7561e-05, 9.9082e-01, 9.8580e-05, 2.4414e-04, 3.0579e-05,
```

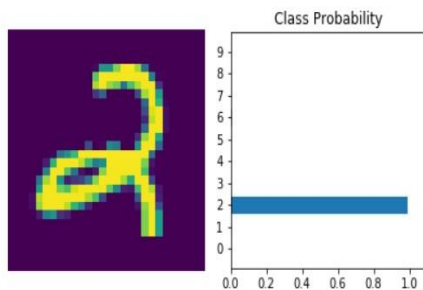
```
In [4]: %matplotlib inline
import helper

images, labels = next(iter(trainloader))

img = images[0].view(1, 784)
# Turn off gradients to speed up this part
with torch.no_grad():
    logits = model.forward(img)

# Output of the network are logits, need to take softmax for probabilities
ps = torch.exp(logits)
helper.view_classify(img.view(1, 28, 28), ps)
print(labels[0])
print(ps)

tensor(2)
tensor([[4.2455e-05, 1.7561e-05, 9.9082e-01, 9.8580e-05, 2.4414e-04, 3.0579e-05,
        6.6148e-03, 9.1356e-08, 2.1302e-03, 6.0150e-06]])
```

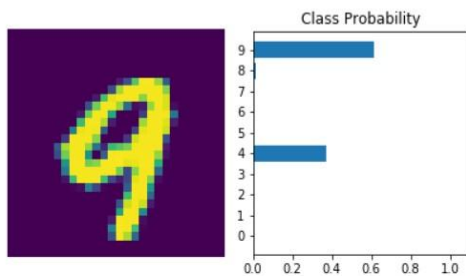


```
images, labels = next(iter(trainloader))

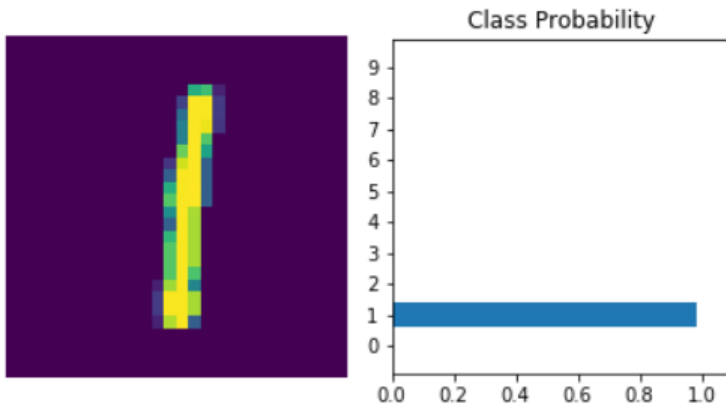
img = images[0].view(1, 784)
# Turn off gradients to speed up this part
with torch.no_grad():
    logits = model.forward(img)

# Output of the network are logits, need to take softmax for probabilities
ps = torch.exp(logits)
helper.view_classify(img.view(1, 28, 28), ps)
print(labels[0])
print(ps)

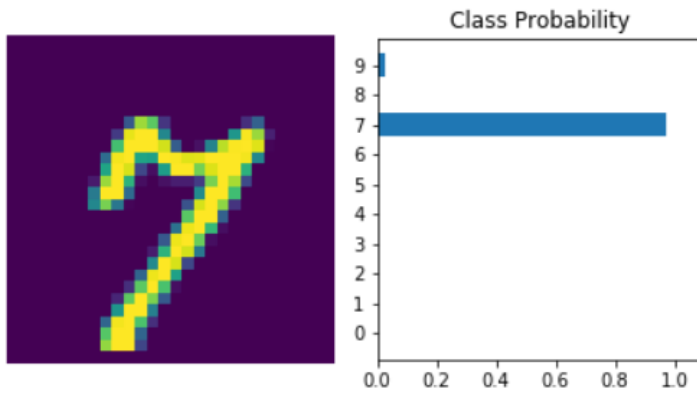
tensor(9)
tensor([[1.8565e-04, 1.1899e-07, 3.0189e-04, 4.9557e-05, 3.7140e-01, 2.9218e-04,
        5.0907e-04, 1.9466e-03, 1.2649e-02, 6.1266e-01]])
```



```
tensor(1)
tensor([[1.1943e-06, 9.8708e-01, 2.3312e-03, 4.9890e-03, 4.5646e-05, 1.3645e-03,
         1.2020e-04, 1.3739e-03, 2.3167e-03, 3.7504e-04]])
```



```
tensor(7)
tensor([[1.3863e-05, 1.8340e-06, 1.6791e-06, 2.1198e-05, 1.3752e-03, 7.8426e-05,
         1.7783e-07, 9.7318e-01, 7.6319e-05, 2.5248e-02]])
```



References

1. IEEE paper: <https://ieeexplore.ieee.org/xpl/mostRecentIssue.jsp?punumber=6973086>
- 2.(Online Resource) <https://www.einfochips.com/blog/understanding-image-recognition-and-its-uses/>
- 3.(Online Resource). <https://medium.com/intro-to-artificial-intelligence/simple-image-classification-using-deep-learning-deep-learning-series-two-5e5b89e979two6>
4. <https://medium.com/@sidereal/cnns-structures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5>
5. Online Resources) <https://skymind.ai/wiki/convolutional-network>
- 6.(Online Resources) <https://udacity.com>
- [7](Online Resources) Khan Academy

ORIGINALITY REPORT

20%
SIMILARITY
INDEX

0%
INTERNET SOURCES

0%
PUBLICATIONS

20%
STUDENT PAPERS

PRIMARY SOURCES



**Submitted to Jaypee University of Information
Technology**
Student Paper

20%

Exclude quotes On

Exclude matches Off

Exclude bibliography On