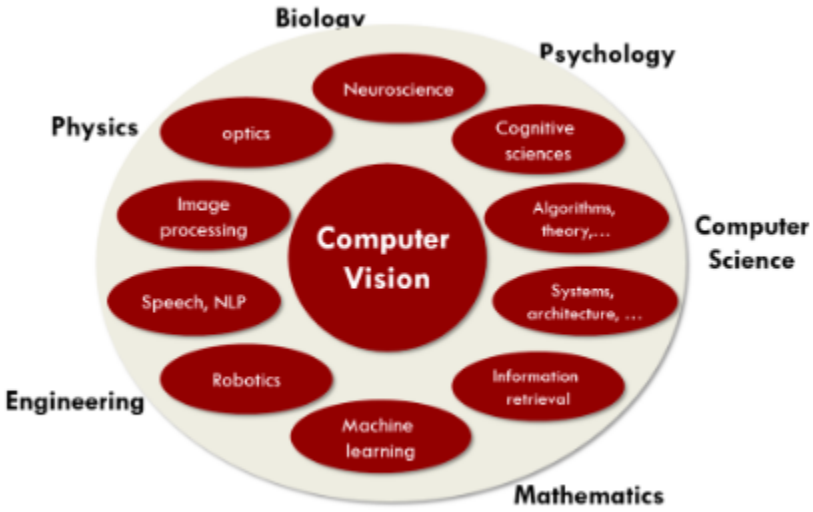
Introduction



Computer vision is an interdisciplinary scientific field that deals with how computers can gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to understand and automate tasks that the human visual system can do.
Sub-domains of computer vision include scene reconstruction, object detection, event detection, video tracking, object recognition, 3D pose estimation, learning, indexing, motion estimation, visual servoing, 3D scene modeling, and image restoration.

Applications of computer vision

Cameras are everywhere and the number of images uploaded on the internet is growing exponentially. We have images on Instagram,videos on YouTube, feeds of security cameras, medical and scientific images... Computer vision is essential because we need to sort through these images and enable computers to understand their content.

Special effects

Shape and motion capture are new techniques used in movies to animate digital characters by recording the movements played by a human actor. In order to do that, we have to find the exact positions of markers on the actor’s face in a 3D space,and then recreate them on the digital avatar.

Scene recognition

It is possible to recognize the location where a photo was taken. For instance, a photo of a landmark can be com-pared to billions of photos on google to find the best matches. We can then identify the best match and deduce the location of the photo.

Face detection

Face detection has been used for multiple years in cameras to take better pictures and focus on the faces. Smile detection can allow a camera to take pictures automatically when the subject is smiling. Face recognition is more difficult than face detection, but with the scale of today's data, companies like Facebook are able to get very good performance. Finally, we can also use computer vision for biometrics, using unique iris pattern recognition or fingerprints.

Optical Character Recognition

One of the oldest successful applications of computer vision is to recognize characters and numbers. This can be used to read zip codes, or license plates. With computer vision, we can do a search on Google using an image as the query.

Self-driving cars

Autonomous driving is one of the hottest applications of computer vision. Companies like Tesla, Google or General Motors compete to be the first to build a fully autonomous car.

Automatic checkout

Amazon Go is a new kind of store that has no checkout. With computer vision, algorithms detect exactly which products you take and they charge you as you walk out of the store.

Vision-based interaction

Microsoft's Kinect captures movement in real time and allows players to interact directly with a game through moves. AR is also a very hot field right now, and multiple companies are competing to provide the best mobile AR platform. Apple released ARKit in June and has already made impressive applications.

Virtual Reality

VR is using similar computer vision techniques as AR. The algorithm needs to know the position of a user, and the positions of all the objects around. As the user moves around,everything needs to be updated in a realistic and smooth way.

Features of Images

Feature extraction is a core component of computer vision. In fact, the entire CV system works around the idea of extracting useful features which clearly define the objects in the image.

A feature in machine learning is an individual measurable property or characteristic of a phenomenon being observed. Features are the input that you feed to your machine learning model to output a prediction or classification. Selecting good features that clearly distinguish your objects increases the predictive power of machine learning algorithms.

In computer vision, a feature is a measurable piece of data in your image which is unique to this specific object. It may be a distinct color in an image or a specific shape such as a line, edge, or an image segment. A good feature is used to distinguish objects from one another. For example, a feature like a wheel enables computers to distinguish between a vehicle and an animal.

Image Types

Grayscale Image Each pixel is a shade of gray with pixel values ranging between 0(black) and255(white).

Color Images have multiple color channels; each color image can be represented in different color models (e.g., RGB, LAB, HSV). For example, an image in the RGB model consists of red, green, and bluechannel. Each pixel in a channel has intensity values ranging from 0-255. Please note that this range depends on the choice of color model. A 3Dtensor usually represents color images (Width x Length x 3), where the 3 channels can represent the color model such as RGB(Red-Green-Blue), LAB (Lightness-A-B), and HSV (Hue-Saturation-Value).

Resolution, defined in dots per inch (DPI), is the number of pixels in an area of inch squared. Higher the DPI, sharper the image

Convolutional Neural Networks

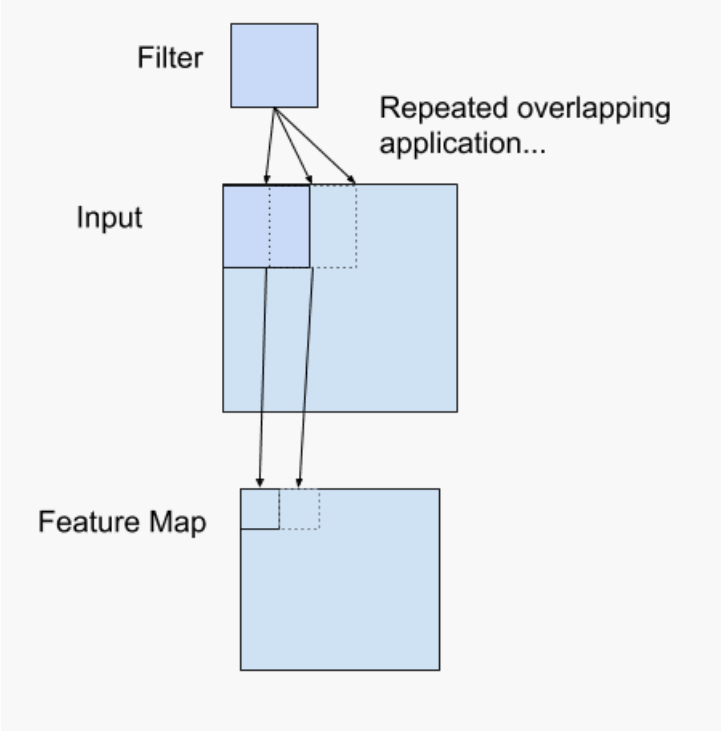
Convolutional Neural Networks, or CNN as they're popularly called, are the go-to deep learning architecture for computer vision tasks, such as object detection, image segmentation, facial recognition, among others. CNNs have even been extended to the field of video analysis

CNN uses filters to extract features from images. It also does so in such a way that position information of pixels is retained.

Convolution Layer

A Convolution layer, the building unit of CNN, performs a mathematical operation called Convolution on 2D image data. The operation was designed for two-dimensional input, the multiplication (Element wise multiplication or dot product) is performed between an array of input data and a two-dimensional array of weights, called a filter or a kernel.

The filter, usually of size 3 x 3 or 2 x 2, is then moved across the image systematically, top to bottom and left to right. The output from multiplying the filter with the input array one time is a single value. As the filter is applied multiple times to the input array, the result is a two-dimensional array of output values that represent a filtering of the input. As such, the two-dimensional output array from this operation is called a "*feature map*".



Previously, the filters were developed by Computer Vision experts (by hand), which made them inefficient and error prone. The innovation of using the convolution operation in a neural network is that the values of the filter are weights to be learned during the training of the network.

The network will learn what types of features to extract from the input. Specifically, training under stochastic gradient descent, the network is forced to learn to extract features from the image that minimize the loss for the specific task the network is being trained to solve, e.g. extract features that are the most useful for classifying images as dogs or cats.

Convolutional neural networks do not learn a single filter; they, in fact, learn multiple features in parallel for a given input. For example, it is common for a convolutional layer to learn from 32 to 512 filters in parallel for a given input. This gives the model 32, or even 512, different ways of extracting features from an input, or many different ways of "*seeing*" the input data.

Pooling Layer

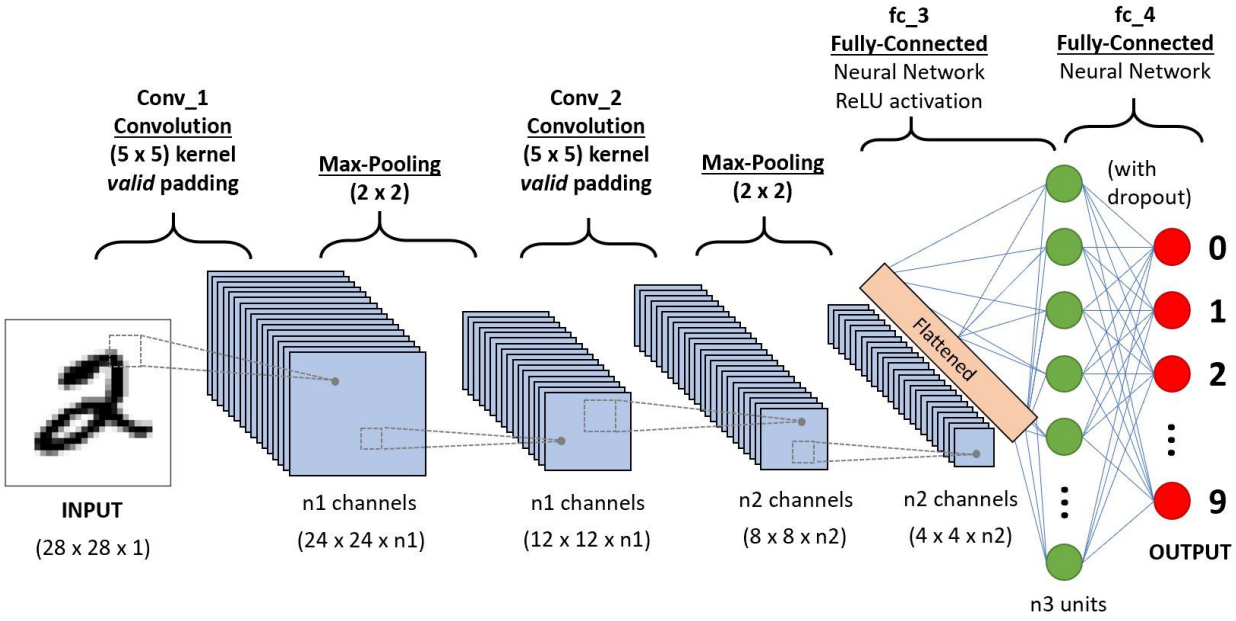
Another power tool that CNNs use is called pooling. Pooling is a way to take large images and shrink them down while preserving the most important information in them. The math behind pooling consists of stepping a small window across an image and taking the maximum value from the window at each step. In practice, a window 2 or 3 pixels on a side and steps of 2 pixels work well.

After pooling, an image has about a quarter as many pixels as it started with. Because it keeps the maximum value from each window, it preserves the best fits of each feature within the window.

A small but important player in this process is the Rectified Linear Unit or ReLU. It's math is also very simple—wherever a negative number occurs, swap it out for a 0.

Next the output is flattened, aka converted from 2D to 1D and fed to a fully connected dense layer, to which the output head is attached.

A typical CNN model looks like this:



The above CNN will be used to predict handwritten digits.

Transfer Learning

Transfer learning generally refers to a process where a model trained on one problem is used in some way on a second related problem. In deep learning, transfer learning is a technique whereby a neural network model is first trained on a problem similar to the problem that is being solved. One or more layers from the trained model are then used in a new model trained on the problem of interest.

Transfer learning has the benefit of decreasing the training time for a neural network model and can result in lower generalization error. The weights in re-used layers may be used as the starting point for the training process and adapted in response to the new problem. This usage treats transfer learning as a type of weight initialization scheme. This may be useful when the first related problem has a lot more labeled data than the problem of interest and the similarity in the structure of the problem may be useful in both contexts.

Models popularly used for transfer learning.

- VGG (e.g. VGG16 or VGG19).
- GoogLeNet (e.g. InceptionV3).
- Residual Network (e.g. ResNet50)

Problem Statement

Problem Statement:
Images are one of the major sources of data in the field of data science and AI. This field is making appropriate use of information that can be gathered through images by examining its features and details. We are trying to give you an exposure of how an end to end project is developed in this field.
The idea behind this project is to build a deep learning-based Image Classification model on images that will be scraped from e-commerce portals. This is done to make the model more and more robust.
This task is divided into two phases: Data Collection and Mode Building.
Data Collection Phase: In this section, you need to scrape images from e-commerce portal, Amazon.com. The clothing categories used for scraping will be:
Sarees (women)
Trousers (men)
Jeans (men)
You need to scrape images of these 3 categories and build your data from it. That data will be provided as an input to your deep learning problem. You need to scrape a minimum 200 images of each category. There is no maximum limit to the data collection. You are free to apply image augmentation techniques to increase the size of your data but make sure the quality of data is not compromised.
Remember, in case of deep learning models, the data needs to be big for building a good performing model. More the data, the better the results.
Model Building Phase: After the data collection and preparation is done, you need to build an image classification model that will classify between these 3 categories mentioned above

Data Collection

We need to scrape 3 types of images off amazon.in: saree, men's trousers, men's jeans. We will be using 2 great python libraries for scraping: Selenium and BeautifulSoup

Selenium

The selenium package is used to automate web browser interaction from Python. Several browsers/drivers are supported (Firefox, Chrome, Internet Explorer), as well as the Remote protocol.
Selenium Scripts are built to do some tedious tasks which can be automated using headless web browsers (Automated Web browsers) . For example, Searching for some Questions on Different Search engines and storing results in a file by visiting each link. This task can take a long for a normal human being but with the help of selenium scripts one can easily do it

BeautifulSoup

Beautiful Soup is a Python package for parsing HTML and XML documents . It creates a parse tree for parsed pages that can be used to extract data from HTML, which is useful for web scraping.

Usually HTMLs are poorly structured and there are numerous obstacles to extract information from it. BeautifulSoup makes it really easy to extract information.

The scraping algorithm

The following steps were followed when

1. Set up Firefox headless browser using Selenium webdriver and Gecko Driver Manager
2. Go to amazon.in, search for the product, and retrieve the page 1 of products page. Parse the page HTML using BeautifulSoup, extract individual product info
3. Open the webpage for individual products, scrape and save the image in a specific folder. Go to next product
4. Once all product images on page 1 are scraped, proceed to page 2.
5. Repeat the process till a threshold no of images are reached.

We scraped a total of 940 images,

- 334 Images of Saree
- 286 images of Trousers
- 320 images of Jeans

The dataset is quite small, but we will apply techniques like Image Augmentation and Transfer Learning to make our model good.

Importing Data and Preprocessing

We will use Google Colab Notebooks with GPU hardware to build and train the model.

About Training with GPU:

As deep learning models spend a large amount of time in training, even powerful CPUs weren't efficient enough to handle many computations at a given time and this is the area where GPUs simply outperformed CPUs due to its parallelism.
A CPU is divided into multiple cores so that they can take on multiple tasks at the same time, whereas GPUs have hundreds and thousands of cores, all of which are dedicated towards a single task. These are simple computations that are performed more frequently and are independent of each other. And both store frequently required data into their respective cache memory.
Neural networks are said to be embarrassingly parallel, which means computations in neural networks can be executed in parallel easily and they are independent of each other.
So most of the tasks are performed with CPU, but when it is time to train the model by calculating weights and back-propagation, the task shifts to GPU.

Importing Data.

Images cannot be directly fed to the CNN, first we have to convert them into proper input format. Also we need to augment the images before feeding them to the model.
We will be using Tensorflow datasets for storing the images .
But first, we need to do some additional steps

Image Augmentation

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

Training deep learning neural network models on more data can result in more skillful models, and the augmentation techniques can create variations of the images that can improve the ability of the fit models to generalize what they have learned to new images.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the *ImageDataGenerator* class.

The following Augmentation techniques will be applied:

Width Shift, Height Shift, zooming image in or out, flipping the image horizontally.

```
#Step 1: Making tensorflow dataset

#Making a generator
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   fill_mode = 'nearest',
                                   rotation_range=60
                                   )

test_datagen = ImageDataGenerator(rescale = 1./255)

#Making tensorflow datast
train_data = train_datagen.flow_from_directory('/content/output/train/',
                                               target_size = (250,350),
                                               batch_size = 16,
                                               class_mode = 'categorical')

test_data = test_datagen.flow_from_directory('/content/output/val',
                                             target_size = (250,350),
                                             batch_size = 16,
                                             class_mode = 'categorical')
```

The augmentation techniques are only applied to the training set.

Rescaling the pixel values

Pixel values are often unsigned integers in the range between 0 and 255. Although these pixel values can be presented directly to neural network models in their raw format, this can result in challenges during modeling, such as in the slower than expected training of the model.

Instead, there can be great benefits in preparing the image pixel values prior to modeling, such as simply scaling pixel values to the range 0-1 to centering and even standardizing the values. This doesn't change the image in any way. Rescaling is done by dividing each pixel value by 255.

Also, model doesn't accept images of different sizes, so will will set a fixed size of 250 x 350 for all the images

Image Data Generator's flow_from_directory() method takes a path of a directory and generates batches of augmented data. ***flow_from_directory()*** assumes:

1. The root directory contains at least two folders one for train and one for the test.
2. The train folder should contain n sub-directories each containing images of respective classes.
3. The test folder should contain a single folder, which stores all test images.

Now it's time to build a model

Model Training and Evaluation

We will be using the EfficientNetB0 model. EfficientNet, first introduced in Tan and Le, 2019 is among the most efficient models (i.e. requiring least FLOPS for inference) that reaches State-of-the-Art accuracy on both imagenet and common image classification transfer learning tasks.

The model structure is as follows:

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnetb0 (Functional)  (None, 8, 11, 1280)       4049571
_________________________________________________________________
flatten_5 (Flatten)          (None, 112640)            0
_________________________________________________________________
dense_10 (Dense)             (None, 256)               28836096
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 3)                 771
=================================================================
Total params: 32,886,438
Trainable params: 32,844,415
Non-trainable params: 42,023
```

We will compile the model with Adam optimizer having learning rate of 0.00001, ‘categorical_crossentropy’ loss, and accuracy as a metric.

We will include 3 callbacks, which execute during model training whenever a certain condition is met. The callbacks are Reducing Learning rate when validation loss becomes constant, Early Stopping when the model validation accuracy decreases instead of increasing. ModelCheckpoint saves the best model to a folder after every epoch

We trained the model for 10 epochs, and got a training accuracy of 92.14% and validation accuracy of 94.18%.

Conclusion

- Images are one of the major sources of data in the field of data science and AI
- Scene reconstruction, object detection, event detection, video tracking, object recognition are some of the applications of Computer Vision
- Convolutional Neural Networks are very successful in executing the applications given above.
- 3 important layers in CNNs are Convolution Layer, Pooling Layer, Fully connected Dense layer
- Selenium and BeautifulSoup are used for web scraping.
- Image Augmentation can be used to add more images when the dataset is small.
- Transfer Learning can be used to train a very complex model trained on a massive dataset and be used for similar kinds of applications easily. Model trained on one problem is used in some way on a second related problem.
- Using EfficientNet model trained on ImageNet dataset, and fine-tuning it for our application, we achieved an accuracy of 94.2%