Notes for presentation

Problem Statement:

Airport ground operations are one of the main causes of late departures. A common mantra in the aviation industry is “The plane is not making any money while it is on the ground”. This statement represents the underlying hurdles faced by the airline industry. There is a need to strive for efficiency, as airlines wish to keep their planes grounded for as short as possible.

Objective:

Build a system which will take video as an input and recognize the ground activities which are bridge connecting, baggage handling, and miscellaneous. To achieve this, we must build a model based on visual-analytical approach that can automatically recognize the activities.

Reason: Selected these categories as it is widely available from YouTube videos.

Data collection:

Data collection was at potential risk at initial phase. We observed the open-source datasets for action recognition like ActivityNet, UCF101 and HMDB51. It has an average of 100 to 150 videos for each action. Downloaded videos from YouTube & performed trimming operation to focus only on the actions we required. Collected around 130 videos for each category.

Dataset:

This gives the diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination condition, etc.

Data preprocessing:

Trimming operation was performed on all videos post the data collection to focus only on the actions we required.

The datasets contain several variable-length videos that are each associated with a semantic label, corresponding to the action being performed in that video.

For single Frame CNN, ConvLSTM, and LRCN model, the trimmed videos are provided as input directly to the model.

For two stream model, the RGB frames and optical frames are extracted from each video and then provided as input to the model.

Data augmentation:

To increase the dataset, we can perform data augmentation. For this, we have extract frames from each video and applied image augmentation technique with the help of albumentations library.

Applied effects like scale rotate, horizontal flip, blur, optical distortion, and saturation to each video.

We have investigated the videos count in other well-known dataset for Action Recognition. We observed that those datasets like ActivityNet, UCF101 and HMDB51 has an average of 100 videos for each action. So, we have decided not to include the augmented dataset for training purpose as we had a good amount of dataset for training.

What does optical flow denote?

Optical flow is the pattern of apparent motion of image objects between two consecutive frames caused by the movement of object or camera.

Eliminates static background & highly focus on the object in motion.

The apparent motion of the brightness patterns is called as optical flow. • The optical flow is a field of 2D vectors and is. defined on the image domain, i.e. at each pixel (x,y) in the image, there is a vector (u(x,y),v(x,y)) giving the apparent displacement at (x,y) per unit time.

Data Labeling:

A categorical label must be assigned to a video clip based upon the action being performed in the clip.

Dump all the videos into corresponding folder structure.

Data Visualization:

We have visualized the data along with labels to get an idea about what classes we are dealing with. For visualization, we picked random videos from each category and visualize the first frame of the selected videos with their associated labels written.

Data exploration:

Extracted information like frame per second, height, width, duration, and frame count. This data has been saved to a csv file for data exploration.

Understand the distribution of the data using describe method

We have visualized the duration distribution for each category.

You can observe that most of the jet bridge connecting dataset has duration distribution around 10 to 60 seconds and for baggage handling, the duration distribution is around 5 to 10 seconds.

Exploration of algorithms:

What is action recognition?

Task of classifying or predicting the activity or action being performed

Deals with a time series classification problem where you need data from a series of timesteps to correctly classify the action being performed in the video.

How does it work?

Uses computer vision techniques to extract the features from the video.

Earlier methods - fixed, hand-crafted features from video such as SIFT descriptors, HoG features, template matching, filter banks, etc.

Recent methods - detect those feature points automatically by a machine.

The ability of CNNs to extract useful features from images created a quick way to learn features. This became a primary area of focus.

Early stage:

Single Frame CNN:

Process frames from a video separately and do averaging to get results.

This network uses single architecture that fuses information from all frames at the last stage. It works by running an image classification model on every single frame of the video and then average all the individual probabilities to get the final probabilities vector.

Architecture:

Creating a simple CNN classification model with two CNN layers. This has got 2 Conv2D layers with 64 filters with kernel size as 3x3 with relu as activation layer.

Followed by this, **BatchNormalization** is added to perform **normalization technique** between the layers of neural network. This is done along mini-batches instead of full dataset. It serves to **speed up the training** and **use higher learning rates**, making **learning easier**. It provides some **regularization and reduces generalization error.**

Then, **Max pooling layer** is added to perform pooling operation that calculates the maximum value for patches of a feature map, and uses it to create **a downsampled feature map**. It is usually used after a convolutional layer. This **highlights the most present feature** in the patch and reduces the spatial size of the representation by **reducing the amount of parameters and computation** in the network. Internally, it reduces the dimensionality of images by reducing the number of pixels in the output from the previous convolutional layer.

Global Average Pooling is added to replace full connected layers in classical CNNs. The idea is to generate **one feature map for each corresponding categor**y of the classification task. The 2D Global average pooling block takes a tensor of size (input width) x (input height) x (input channels) and computes the average value of all values across the entire (input width) x (input height) matrix for each of the (input channels).

ReLU sets all negative values in the matrix x to zero and all other values are kept constant. ReLU is computed after the convolution and is a nonlinear activation function like tanh or sigmoid. More computationally efficient to compute than Sigmoid. Just needs to pick max(0, x) and not perform expensive exponential operations. Relu tend to show **better convergence performance** than sigmoid.

Softmax activation function in the final output layer when you are trying to solve the Classification problems where your labels are class values. It predicts a **multinomial probability distribution** - for multi-class classification problems. A Softmax function is a type of **squashing function**. Squashing functions limit the output of the function into the **range 0 to 1**.

Disadvantage:

ignoring the time-based changes in the actions

only try to learn the environmental context with the help of multiple frames

The results will change rapidly and fluctuate

model is not looking at the entire video sequence but just classifying each frame independently.

Another methodology

which was introduced to address the problem of action recognition. This uses spatio-temporal features of video. This will train a model based on features at various time stamps. This ensures that we consider the sequence of action being performed.

The action being performed is dependent on the sequence of activities performed at various time intervals.

CNN+LSTM:

CNN are great for image data and LSTM network are great when working with sequence data.

The video was first segmented into smaller groups of frames which are passed into CNN to extract features. However, instead of performing classification on each of these features independently, the features associated with each video segment are passed, in temporal order, as input to an LSTM, which then performs the final classification.

This has shown to consider the context of the full video and, as a result, achieve significantly improved performance

Experimenting two architecture that will be using CNN along with LSTM. They are

1. ConvLSTM
2. RCN

ConvLSTM Network:

This is implemented by using a combination of ConvLSTM cells. A ConvLSTM cell is a variant of an LSTM network. It has got convolution embedded in the architecture, which makes it capable of identifying spatial features of the data while keeping into account the temporal relation.

works effectively for video classification problem as it captures spatial relation in the individual frames and the temporal relations across the different frames. This ConvLSTM's convolution structure can take 3D input (width, height, num_of_channels) whereas a simple LSTM only takes 1D input hence an LSTM is incompatible for modeling spatio-temporal data on its own.

Architecture:

3 ConvLSTM cells are being used

We use **Keras convLSTM2D** recurrent layers to construct the model. This convLSTM2D layer takes **number of filters and kernel size** as input for performing convolutional operations.

MaxPooling3D layers to r**educe the dimensions** of the frames and avoid unnecessary computations.

The **dropout layers** are added at each layer to **prevent the overfitting problem.**

The output layer is flattened and fed to the Dense layer with **softmax** as an activation which outputs the probability of each action category.

LRCN Network: (Long Recurrent Convolution Network)

Combines CNN and LSTM layers in a single model.

The CNN model can be used to extract spatial features from the frames in the video and LSTM model can then use the spatial features extracted by CNN at each time-steps for temporal sequence modeling.

Architecture:

Uses **time-distributed conv2D** layers, which allows **applying the same layer to every frame** of the video independently. This layer can take input of shape (no_of_frames, width, height, num_of_channels). This is very beneficial as it allows to input the whole video into the model in a single shot.

This layer is followed by MaxPooling2D and Dropout layers.

The spatial features extracted from the **Conv2D layers** will be flattened using Flatten layer. This flatten output is then passed onto LSTM layer.

The output of LSTM layer is fed into the Dense layer with **softmax activation** which will be used to predict the action being performed.

Two Stream Network:

Composed of **two separate convolution neural networks**.

One to handle spatial features and one to handle temporal or motion features.

The input to the two-stream architecture only contains a **single frame for the spatial stream** and a fixed-**size group of optical flow maps** for the temporal stream.

The output of these separate network components can be combined to form a spatiotemporal video representation.

L**ong term dependency** is modeled by combining temporal network output across time frames.

```
def __init__(self, block, layers, num_classes=1000):
    self.inplanes = 64
    super(ResNet, self).__init__()
    self.conv1 = nn.Conv2d(3, 64, kernel_size=7, stride=2, padding=3,
                           bias=False)
    self.bn1 = nn.BatchNorm2d(64)
    self.relu = nn.ReLU(inplace=True)
    self.maxpool = nn.MaxPool2d(kernel_size=3, stride=2, padding=1)
    self.layer1 = self._make_layer(block, 64, layers[0])
    self.layer2 = self._make_layer(block, 128, layers[1], stride=2)
    self.layer3 = self._make_layer(block, 256, layers[2], stride=2)
    self.layer4 = self._make_layer(block, 512, layers[3], stride=2)
    self.avgpool = nn.AvgPool2d(7)
    # self.fc_aux = nn.Linear(512 * block.expansion, 101)
    self.dp = nn.Dropout(p=0.8)
    self.fc_action = nn.Linear(512 * block.expansion, num_classes)
    # self.bn_final = nn.BatchNorm1d(num_classes)
    # self.fc2 = nn.Linear(num_classes, num_classes)
    # self.fc_final = nn.Linear(num_classes, 101)
```

Challenges faced:

Procuring the data sets has been a big challenging task as there were limited dataset available on the internet. The dataset was collected from YouTube. After downloading, we have performed trimming operation to focus only on the actions we required. This way we only have only one action category in each video

Size of training and testing data: 80% and 20%

Why this algorithm is chosen vs other and conclusion:

From the model training results, you can see the LRCN model is performing well in terms of loss and accuracy. Also, you can visualize the result of model inference. Both training and testing results are good in LRCN model. Thus, I conclude that we can chose this model to identify the gate operations

Accuracy of the solution:

Evaluation metric:

Definition:

The confidence score is a number between 0 and 1. A score of 1 is likely an exact match, while a score 0 means, that no matching answer was found. The highest score, the greater the confidence in the answer.

What it signifies?

This score shows the probability of the image being detected correctly by the algorithm and is given as a percentage. Higher confidence scores indicate greater probable accuracy, while lower confidence scores indicate less probable accuracy

How is it calculated?

The scores are taken on the mean average precision at different IoU (Intersection over Union) thresholds.

Intersection over Union (IoU) is measured by the magnitude of overlap between two bounding boxes of two objects. The formula of IoU is given below:

Precision is calculated at the threshold value depending on the True Positives (TP), False Positives (FP), and False Negatives (FN) which are the result of comparing the predicted object to the ground truth object

Precision = TruePositives / (TruePositives + FalsePositives)

What is Accuracy Formula? The accuracy formula provides accuracy as **a difference of error rate from 100%**. To find accuracy we first need to calculate the error rate. And the error rate is the percentage value of the difference of the observed and the actual value, divided by the actual value.

Future work:

This model output can be later used to calculate the time taken by each action is observed using the start and stop instances of the predicted action and these time values are used for analysis and optimization of scheduling operations in future. The display feature of historical data can be used to estimate their completion time and plan the activities more precisely and thereby, enhance passenger experience. With this model, occupancy rates and processing times can be tracked and optimized. Safety and security incidents can also be recorded for investigation and hindsight. By this way, we can improve the degree of complexity that can be handled by existing real-time surveillance systems.

Challenges like overfitting and underfitting:

A model that exhibits small variance and high bias will underfit the target, while a model with high variance and little bias will overfit the target.

Organization usage:

While stationary, the airline is not earning any revenue with its plane, while facing many costs at the same time. Not having a quick turnaround time can cost business. In many cases, the amount of time it takes to complete represents cost and can present risk. Fast turnaround time means less money spent on manpower.

Lessons Learnt

Code walkthrough and demo

Test the model