Nile University
جامعة النيل

Selected Topics in Deep Learning

Human Pose Detection Project

Final Project Based on MPII Dataset

Submitted to:
Prof. Dr. Mohammed El-Beltagy

Submitted by:
Khaled Adel Ezzat
Riham Mohamed Nour El-Din
Mohamed Hussein
Amr Medhat

Introduction:

State the problem and what you are trying to accomplish

Human Pose Detection Project Based on MPII Dataset which provides images data of humans with different poses, taken from YouTube videos, and their detailed information and annotations with Category and Activity included in annotation file

Task:

The objective of this project is to create a Training Model to understand and provide a classification for the images given and another bonus Training Model that provides further description about the activity that the human is doing in the image in ease the Model should train on training dataset then to be able to predict the Category and the Activities of any unseen image data associated with it.

Introduction

This report has been done to build a deep learning model capable of detecting human activities from images. The dataset used is the MPII Human Pose Dataset available online. The approach used included training a deep learning model utilizing multiple pretrained models to have better training and achieve better results. The pretrained models used in this report are Xception, ResNet50, VGG16 and VGG19. Model used were Keras Sequential models using one of the pretrained models as a first layer, then stacking up on various layers seeking different results. The model using the Xception pretrained model shower the highest training validation percentages reaching up to 100 percent, but with lower validation accuracy, the other models showed training results in the range of the 90s, still with lower validation accuracies.

Data Extraction

The dataset used in this project is the MPII Human Pose Dataset, which is composed of around 25K images containing over 40K people with annotated body joints. The dataset covers 410 human activities and each image is provided with an activity label. Each image was extracted from a YouTube video.

The data extraction process included downloading the compressed data files, extracting the images, obtaining annotations from the compressed annotations file, and extracting the corresponding activity to each image. Afterwards data filtering was done to split the data into training and testing data for the model fitting and evaluation.

After data download and extraction, image generation and data augmentation have been done on the images in order to do data preprocessing for every pretrained model, adding data

augmentation using a rotation degree, and resizing images for the model. Also, data generation has been used in order to split the training data into training and validation data with a fraction between 20 and 25 percent of the total training data.

In order to save memory, data has been used using the !wget command to form pull it from the site provided instead of downloading the data then uploading it again.

Data Exploration

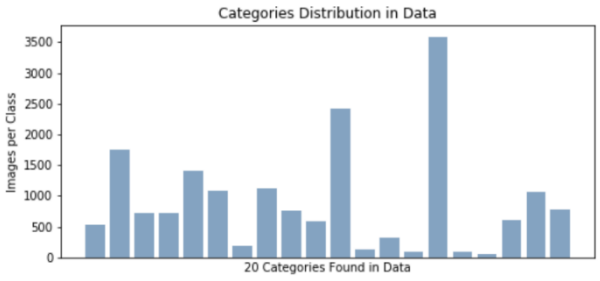
- Data Inspection

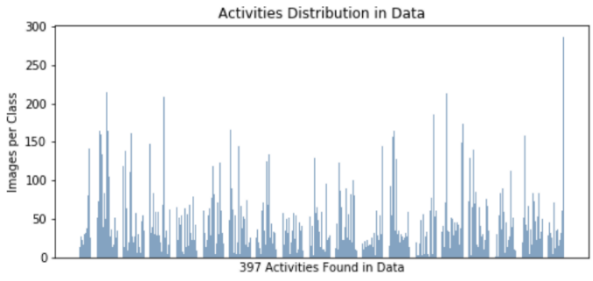
The annotation file provided we extracted the needed data for our task which contains 24987 records of all provided data which are spitted into Training and Testing images based on column Train which is by default 1 which are 18079 training record and 0 for 6908 testing records, and here is a snippet for the data after extraction is:

	Image	Category	Activity	Train
0	053686627.jpg	NaN	NaN	0.0
1	095071431.jpg	transportation	NaN	0.0
2	073199394.jpg	NaN	pushing car	0.0
3	059865848.jpg	NaN	NaN	0.0
4	015601864.jpg	sports	curling	1.0

- Data Visualizations

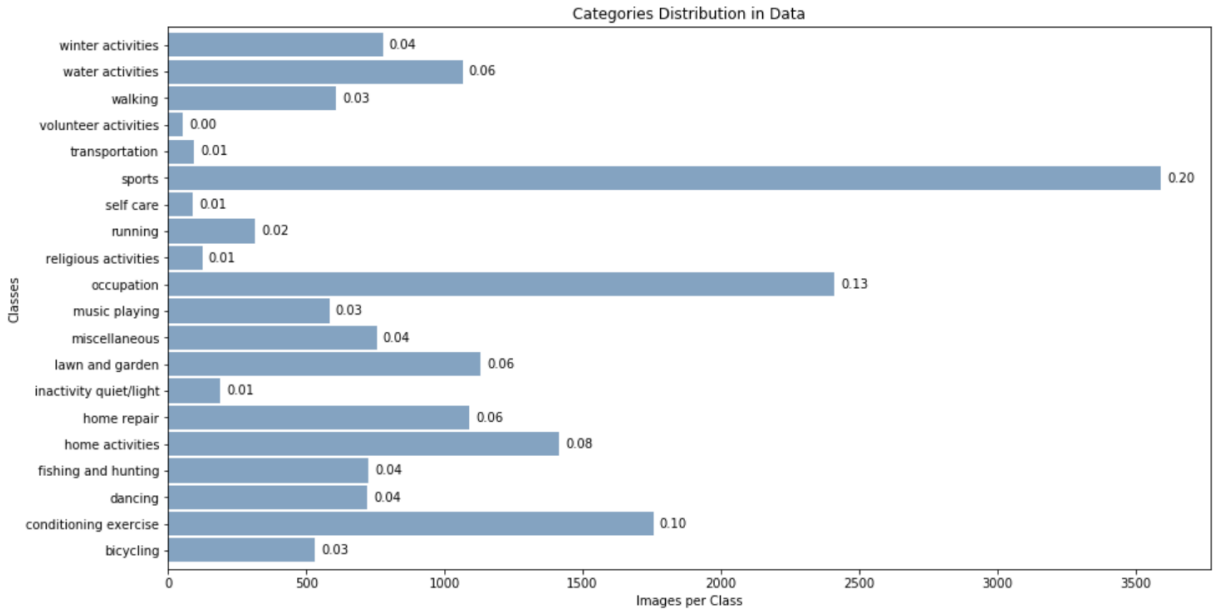
The provided data consists of 20 unique categories for all images and 397 different activities:





- Data Imbalance

Data plotting describes that the data have unbalanced classes as shown which affects the model to bias towards some classes this will be handled into the models when we get to it.



Regarding the category classes, the following plot shows that some categories are found with much higher frequencies than others, which might cause a bias while training or testing. To overcome this, a class_weights parameter is added to overcome this.

Deep Learning Models

In order to be able to detect a human activity from an image, a deep learning model has been built, compiled and fitted to training and validation data in order to be able to predict the activities of given images.

For this project, all models used are Keras Sequential models, using one of multiple pretrained models as the first layer, then building up other Keras Flatten, Dense, and Dropout layers.

The various models used to experiment the data with in this project are as followed

Xception Model

In this approach, a sequential model was built using an Xception pretrained model as the first layer, then flattening was applied on the data followed by a dense layer using the Softmax activation for classification. The Xception layer had its last 7 layers untrainable, in order to replace the training with the MPII dataset used for this project.

The model used the Adam optimizer, and has been trained for 30 epochs achieving a training accuracy of 100%, validation accuracy of 80%, whereas the training loss was 0% along with a validation loss of 0.8193%.

- Data tuning included the following:

- Regarding the layers of the model, two approaches were experimented:

    1. The first attempt had the stack of the model made of the Xception pretrained model layer, Flatten layer, Dense layer with relu activation, Dropout layer, then finally Dense layer with Softmax activation for classification.

    2. The second attempt had the stack of the model made of Xception model, Flatten layer, then finally Dense layer with Softmax activation for classification. That approach showed better results regarding training and validation accuracies.

- Regarding the batch size, 30 and 100 batch sizes were used and 100 showed better results.

- When choosing the compilation optimizer, the following optimizers were used:

    1. SGD
    2. SGD (Nesterov)
    3. Adam. This optimizer showed better results

- Deciding on the trainable layers of the model included two approaches:

    1. Making the whole Xception layer untrainable

    2. Making the last layers of the Xception layer untrainable. In that approach, we experimented with three numbers of untrainable layers, the last 7, 17 or 47 layers. The best result was obtained while making the last 7 or 47 layers untrainable.

- Data has either been shuffled or not before image generation, and shuffling has improved the results of the model.

- Data augmentation has been done using various approaches including the zoom range, sheer range, and horizontal flip. Also, tuning included either using the augmented data alone, or appending the augmented data to the original data, but the latter didn’t show good results.

- Model testing using this pretrained model yielded an accuracy of 100%.

ResNet50 Model

In this approach, a sequential model was built using a ResNet50 pretrained model as the first layer, then flattening was applied on the data followed by a dense layer using the Softmax activation for classification. The ResNet50 layer had its last 10 layers untrainable, in order to replace the training with the MPII dataset used for this project.

The model used the Adam optimizer, and has been trained for 30 epochs achieving a training accuracy of 97%, validation accuracy of 40%.

- Data tuning included the following:

- Regarding the layers of the model, two approaches were experimented:

    1. The first attempt had the stack of the model made of the ResNet50 pretrained model layer, Flatten layer, Dense layer with relu activation, Dropout layer, then finally Dense layer with Softmax activation for classification.

    2. The second attempt had the stack of the model made of ResNet50 model, Flatten layer, then finally Dense layer with Softmax activation for classification. That approach showed better results regarding training and validation accuracies.

- Regarding the batch size, 30 and 100 batch sizes were used and 100 showed better results.

- When choosing the compilation optimizer, the following optimizers were used:

    1. SGD
    2. Adam. This optimizer showed better results

- Deciding on the trainable layers of the model included two approaches:

    1. Making the whole ResNet50 layer untrainable

    2. Making the last layers of the ResNet50 layer untrainable. In that approach, we experimented with two numbers of untrainable layers, the last 10 or 20layers. The best result was obtained while making the last 10 layers untrainable.

- Data has either been shuffled or not before image generation, and shuffling has improved the results of the model.

- Data augmentation has been done using rotation angle. Model testing using this pretrained model yielded an accuracy of 83%.

VGG16 Model

In this approach, a sequential model was built using a VGG16 pretrained model as the first layer, then flattening was applied on the data followed by a dense layer using the Softmax activation for classification. The VGG16 layer had its last 10 layers untrainable, in order to replace the training with the MPII dataset used for this project.

The model used the Adam optimizer, and has been trained for 30 epochs achieving a training accuracy of 98%, validation accuracy of 72%, whereas the training loss was 0.033 along with a validation loss of 1.533

- Data tuning included the following:

- Regarding the layers of the model, two approaches were experimented:

    1. The first attempt had the stack of the model made of the VGG16 pretrained model layer, Flatten layer, Dense layer with relu activation, Dropout layer, then finally Dense layer with Softmax activation for classification.

    2. The second attempt had the stack of the model made of VGG16 model, Flatten layer, then finally Dense layer with Softmax activation for classification. That approach showed better results regarding training and validation accuracies.

- Regarding the batch size, 30 and 100 batch sizes were used and 100 showed better results.

- When choosing the compilation optimizer, the following optimizers were used:

    1. SGD
    2. Adam. This optimizer showed better results

- Deciding on the trainable layers of the model included two approaches:

    1. Making the whole VGG16 layer untrainable

    2. Making the last layers of the VGG16 layer untrainable. In that approach, we experimented with two numbers of untrainable layers, the last 10 or 20 layers. The best result was obtained while making the last 10 layers untrainable.

- Data has either been shuffled or not before image generation, and shuffling has improved the results of the model.

InceptionResNetV2 Model

In this approach, a sequential model was built using an InceptionResNetV2 pretrained model as the first layer, then flattening was applied on the data followed by a dropout layer, then a dense layer using the Softmax activation for classification.

The model used the Adam optimizer, and has been trained for 10 epochs with a batc size of 100 achieving a training accuracy of 94%, validation accuracy of 65%, whereas the training loss was 0.3 along with a validation loss of 6.6%

- Data tuning included the following:

- Regarding the layers of the model, the stack of the model is made of the InceptionResNetV2 pretrained model layer, Flatten layer, Dropout layer, then finally a Dense layer with Softmax activation for classification.

- Regarding the batch size, 30 and 100 batch sizes were used and 100 showed better results.

- When choosing the compilation optimizer, the Adam optimizer was used.

- Data has either been shuffled before image generation.

Results

- Many Models were trained to optimize and get those results.

Models Exploration							
		Epochs	Optimizer	Batch Size	Freezing Layers	Train Acc	Val Acc
1st attempt	VGG16	10	Adam	50	[-10]	93%	73%
	InceptionResNetV2	10	Adam	100	-	93%	64%
	Xception	10	Adam	30	[-7]	99%	74%
2nd attempt	VGG16	30	Adam	50	[-10]	98%	72%
	InceptionResNetV2	10	Adam	100	-	94%	65%
	Xception	30	Adam	30	[-47]	100%	81%

Parameters Exploration						
	Epochs	Batch Size	Freezing Layers	Optimizer	Data Shuffling	Accuracy
Xception	30	30	[-7]	Adam	True	86%
Xception	30	100	[-7]	Adam	True	99%
Xception	30	100	[-7]	SGD (Nesterov = True)	True	78%
Xception	30	100	[-47]	SGD (Nesterov = True)	True	99%
Xception	30	100	[-47]	Adam	True	100%

Testing

In this stage the best Models in the Paramters Expermentation phase were selected to test on, Testing Results were nearly identical to validation accuracy, with margins of 4, Three Models was in this stage, Two Xceptions and One InceptionResNetv2 achiving accuracies: 83.5%, 72.9%, and 67.2%.

1. Xception | 30 Epochs | 100 Batch | Adam | -7 --> 83.5%
2. Xception | 30 Epochs | 100 Batch | Neterov | -47 --> 72.9%
3. InceptionResNetV2 | 10 Epochs | 100 Batch | SGD --> 67.2%

Bonus Part

Human Activity detection using Multilabel Classification

Using the MPII Human Pose dataset, multilabel classification has been done on the data in order to detect a human activity from an image.

The image activities are in the form of multiple activities, comma separated, and the trained model used has been done to be able to detect multiple labels.

- Two approaches have been experimented in this part for training:

- Normal classification, and this one has been experimented with two tuning combinations:
    1. SGD optimizer, and a batch size of 100. This yielded an accuracy of 99.91%, validation accuracy of 56.66% along with a test accuracy of 7.8%.
    2. Adam optimizer, and a batch size of 100. This yielded an accuracy of 95.24%, validation accuracy of 47.75% along with a test accuracy of %.
- Multilabel classification, and this approach had a batch size of 100, 30 epochs, and Adam optimizer. This approach has yielded an accuracy of 64.4%.

- Not the worst result, but we expected it to be so, the Model is trying to train on 16,000 image with 600 labels, not that much would train a mode, its 2.6 images to train to classify 1 image, too hard for the mode.

- Limitations

- The Notebook was run at Google with GPU backend 25 Gigs and Storage of 82 Gigs, which was pretty fast and powerful machine, but with Neural Networks its comes also to some limitations:

    1. Augmenting the images then saving them couldn't be done in just 1 batch, we had to split them into smaller batches to be done, with a Error margin of [+ or –] Image or two per batch, so if it were done in one batch 180 Errors could be just one error.
    2. Saving augmented images was a challenge too, as we could only make 1 replica of each images to get total of 36,000 images, which also were fairly small training data comparing them with one million and half the Xception model was trained to, which lead in most of the models to overfitting, so more storage more images more validation error to decrease.
    3. Time took to train the model was roughly long, about 5-6 hours to train 10 Epochs model, so we think if we had more computationally power, we would done more in less time.
    4. Cross validation it had to be done because we don't have that much of data to split some into validation section, Cross validation also doesn't supported by keras, but it has its own implementation with Skit-learn, enough resources, we can make it work.
    5. Most of the cases when loading images to memory lead to crash of memory, and restarting the whole process and models again, with time we had to convert all processes to tensors and process them in parallel, makes it faster, less power, low memory.
    6. For Scaling this dataset with bigger one, it would be perfect for training and prediction but it needs more power at least with more storage, prefer to get multiple nodes with TensorFlow distributed on those nodes for best result, it can be done on existing machine to but it will be tricky somehow as everything has to be made in parallel to not failing in memory trap.
    7. Limited number of images, too few images cannot build the greatest systems at all.
    8. Annotation file format and structure was also challenging or could be a limitation too, it could be a lot easier.

References

- Wikipedia
- Google Colab
- Medium
- Github
- Keras
- TowardsDataScience
- StackOverflow
- Models papers