Deep Learning Course Assignment

Zihadul Azam
Mat. 221747
Zihadul.azam@studenti.unitn.it

Ali Hamza
Mat. 225088
ali.hamza@studenti.unitn.it

Sabbavarapu Hanusha
Mat. 225018
Sabbavarapu.hanusha@studenti.unitn.it

Abstract

Over the past few decades, deep learning has played a vital role in the revolution of the computer vision domain. The following work explores different ways to utilize deep networks to solve real word problems from the domain of the computer vision using the PyTorch framework. The images dataset used was collected from surveillance camera where each image is annotated specifying key attributes of the person in it like clothing color, hair color, age and etc. The work is mainly divided into two part where first part proposes a solution in form of a multi-class classifier to predict above described attributes for images. While second part focuses on the construction of a model for the re-identification of a person when given a query image.

1. Introduction

Deep Learning has given enough power to the machines to solve unimaginably complex and critical problems belonging to various aspects of our daily lives. Among its many other usages, Deep Learning has wide range of applications in the domain of Computer Vision. Many difficult problems like image classification, image segmentation and object detection can be solved with much ease as compared to the traditional Computer Vision techniques. Using ample amount of computing resources and huge data, Deep Learning methods involving deep networks like Residual Neural Networks (RNNs) and Convolutional Neural Networks (CNNs) have reasonably improved performance in terms of prediction accuracy and execution time. Problems that were considered impossible to solve are not only being solved but will astonishing precision and accuracy [1].

During the recent years, Some mentionable achievements in the field of Computer Vision like massive success in the image classification and recognition was influenced by development of CNNs [2]. The mentioned breakthrough was mainly because of tremendous increase in the computational power and the availability of the huge amount of data for the training of these complex neural networks. This all together gave boost to the usage of architectures based on Deep Neural Networks (DNNs) which is also supported by the fact that seminal paper ImageNet Classification with Deep CNNs has more than three thousand citations [3].

Training complex DNNs requires huge amount of data with annotations and can take many hours or even days while training depending upon the computational power of the machine used. Optimization of such models based on the performance and error sometimes requires many iterations of this extremely costly training process in terms of time and computational power. Such models are supposed to work properly where training and test data belongs to same feature space. Any change in the data distribution or feature space might need new model construction. Training a new model every time might seem like waste of time and resources. This is where techniques like transfer learning prove to be very useful [4].

Transfer learning introduces idea of using pre-trained model for solving new or similar problems from the same domain. It appreciates the application of previously learned knowledge while solving a new problem or task [5]. The ImageNet Challenge has been center of attraction for the people working on the image recognition and classification. Many brilliant minds working on it have come up with some state-of-the-art solution/models like **Resnet50**. Mentioned model was able to classify around 1 million images into 1 thousand categories most accurately.

Since the purpose of the first part of the work was almost the same as the ImageNet challenge, so we decided to build upon existing state-of-the-art solutions of ImageNet challenge, for our tasks, exploiting the idea of transfer learning. Instead of starting from scratch, we opted to use Resnet50 of the shelf for the multi-class image classification part of our work. Since the Resnet50 predicts among 1000 categories for the each image, challenging part was to customize it according to our needs; to be able to predict among multiple classes for each attribute like hair color, gender, clothing color and etc.

For the second part of the work, which was about person re-identification, we used the same model trained for the first task and did some modification to it; removed the final output layer because we wanted to extract the features of the image instead of its class and added only one FC layer to predict the person ID. We used **Mean average precision** (mAP) for the evaluation in this task. Mean average precision (mAP) is a simple number obtained by summing the Average Precision scores calculated for each single query [9]. This number is used to quantify performance of a model for such queries, typically used in identification and

detection tasks.

2. Proposed Solution

We started working using pre-trained Resnet50 model and did necessary modification in its architectures to alter the output as per our needs.

2.1. Data Exploration

Real life data usually does not come in ready to feed format and requires quite a lot of pre-processing before feeding to the network. For better understanding of the multi-class classification problem, the available data was explored in detail. Annotations were available for each training image. Annotations data file had following columns.

Id, age, backpack, bag, handbag, clothes, down, up, hair, hat, gender, upblack, upwhite, upred, uppurple, upyellow, upgray, upblue, upgreen, downblack, downwhite, downpink, downpurple, downyellow, downgray, downblue, downgreen, downbrown, filename

2.1.1 Data Pre-processing

After data exploration, it was found that there was a lot of pre-processing needed to make the data ready for training. Python's pandas library was used for this purpose.

Initial step was to map each training image to the provided annotations, for training, in form of a CSV file. All images from the train folder of the dataset were read using pandas. Each image name had PersonID concatenated with "_" which was extracted and stored in new column of the first data-frame as shown below.

	id	filename
0	1059	1059_c6_014145176.jpg
1	466	0466_c5_020049087.jpg

Figure 1

Annotations were read as a new pandas data-frame and later merged with the first data-frame based on id column. Merged data frame now had image file name also along with the annotations for each of its attributes.

Next step was to convert the values of the attributes into binary form. To achieve this, each column of the data-frame except the "id" and "filename" was simply reduced by one. In the provided data annotations, all of the up color and down color columns were zero where the person was earning multi-color clothing on upper body and multi-color bottoms respectively. To handle this case, two new columns named "upmulticolor" and "downmulticolor" were introduced and set to 1 where sum of all up color columns and all down color columns was zero, respectively.

Complex and enormously big dataset usually take a lot more time in training as compared to the small but equally informative datasets. In this dataset, Scikit-learn library's label encoder was used to label all colors for both top and bottom clothes [6]. Instead of using all columns, only label index of the color was stored separately for upper and lower body clothing colors. This technique massively reduced the size of the dataset. At this point, pre-processing of the data was done. Pre-processed annotations data was stored in a new CSV file.

2.1.2 Data Transformation

Deep learning models sometimes tend to overfit over the training data which badly affects the generalization ability of the model. To avoid such issues, data augmentation technique is commonly used. It is used to artificially enlarge the existing dataset by adding noise to existing data, cropping and rotation of the images. Such procedures should not require huge computation and extra storage space to serve the purpose efficiently [1].

We normalized the images data channel-wise with standard ImageNet normalization parameters provided for mean and standard deviation [7]. Torchvision library was use to do the normalization and the transformations like rotation, flipping, erasing and cropping the images randomly.

2.2. Implementation Details

2.2.1 The Model Architecture

We used pre-trained version of **Resnet50** for the multi-class classification part of this work because of being one of the most efficient solution for the image classification. As a base model, we took all layers of the Resnet50 except the final output layer since it does not coincide with our required output.

We first defined custom output classes for all attributes that we wanted to predict. We crated separate classifiers for all of the classes in our custom output layer. This was done using "models" module of the Torchvision library. Final layer takes input from the previous layer. **ReLU** was used as an activation function in the output layer as it provides quite high classification accuracy as compared to the other candidate activation functions. Intuition behind using ReLU is that it perfectly mimics the behavior of biological neurons and tends to provide great accuracy. A custom class named "CustomResNet50" was created for this purpose.

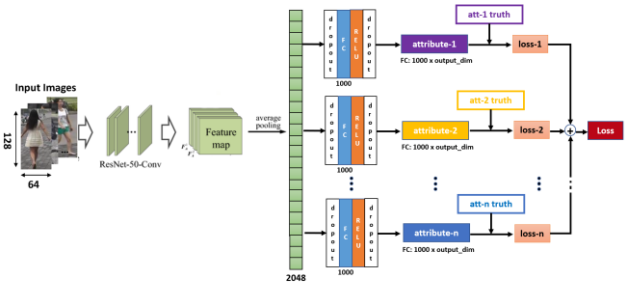

Figure 2 - Classification Model Architecture

For the person re-identification task, have tried three different approaches. The first idea is a very simple solution, take the used model for the first task (classification), remove the classification layers such that we can get the feature values if we feed an image as input to the model. Then, we got feature vectors for each of the query images and the gallery images (test images).

Once we had all the feature vectors, we have calculated the similarity between the query images and the gallery images using **cosine-similarity** function. This is a very simple and time saving solution, because it does not need any further training process for the second task. The model was evaluated in using the Mean Average Precision (mAP) metric. But the produced mAP score was not that bad, but also not good enough, only **65.61 .**, so we decided to go for the solution based on triplet loss.

In this improvement phase, We trained our model by using a joint loss. The joint loss is defined as following:

$$\text{joint\_loss} = \text{triplet\_loss} + \text{id\_loss}$$

Where, triplet loss is the classical triplet loss and the classification loss is the loss of "image_id" prediction, which has been calculated using "cross_entropy" loss function. Anyway, we tried also to calculate the mAP score with only the triplet loss, we trained a model in a similar way of the joint loss approach, but instead of joint loss we used only the triplet loss. The mAP score was smaller then the joint loss mAP score, and this was expected. Because, triplet loss is used to calculate the distance of the sample features, increasing the distance between the anchor and negative sample, reducing the distance between the anchor and the positive sample. Softmax loss (ID loss) performs label-level supervision and constraint on the feature extraction network (see Figure 3 – ReID Model Architecture).

2.2.2 Training-Validation Data Split

The Market-1501 dataset contains 19,732 images for 751 identities for training and 13,328 images for 750 identities for testing. For each image, 27 attributes are annotated. To validate our model we used 651 identities in training set for training and the other 100 identities used as the validation set.

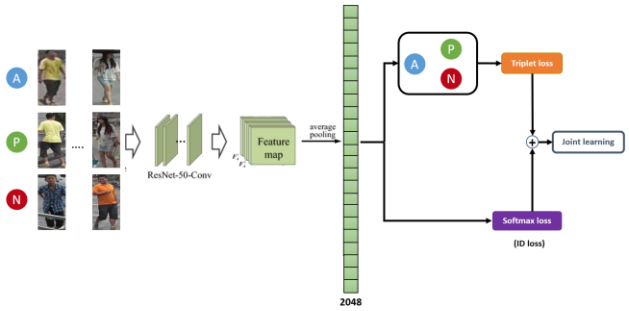

Figure 3 – ReID Model Architecture

2.2.3 Model Training and Validation

After having done the data split for the training and validation, model was trained on the training data. Training was carried out in epochs. DataLoader provided by Torch library are used to iterate over dataset.

A helper function named "move_to" was defined to load data to the device, GPU or CRU based on availability. Model output in form of predictions is pushed to the device for the further processing. "get_loss" helper function was defined to calculate and sum losses of all classifiers using the provided cost function. Stochastic Gradient Descent (SGD) is used as the optimizer in the network. Loss is propagated in the networks backwards and weights of the network are updated by taking optimizer step also resetting the gradient to zero. Total training loss is also updated after the optimization step which is used in the calculation of batch loss.

Training history of the model is compiled. Helper function named "get_all_classifiers_accuracies()" and "calculate_accumulative_accuracy()" were defined to calculate overall accuracy.

Almost similar process is repeated for the validation of the model along few additional steps. Optimization steps like back propagation and gradient reset are not performed during the validation phase. Validation loss and accuracy are calculated after model gives prediction for the validation dataset.

2.2.4 Regularization

While learning, a neural network model tries to update the weights based on the training data. Following this behavior, a model might end up memorizing the training data and cannot perform well on the unseen data showing lack of learning. This property of machine learning and deep learning models is called overfitting. Regularization techniques are used to avoid this problem.

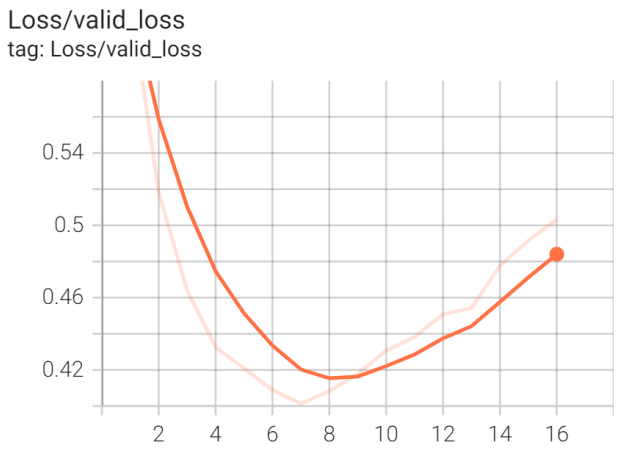

Figure 4

Early stopping is one of the regularization techniques that has been used in this work. In the validation phase, training of the model is stopped as soon as there is no improvement in the validation loss. This process is known as early stopping. This techniques helps saving a lot of time and processing power and give a model that performs relatively better [8].

In the following work, minimum number of epoch to stop training was 10. *Figure 3* shows that the model was not able to improve validation loss after the 7th epoch and was eventually stopped after 17th iteration. Best performing model was saved at the checkpoint path.

2.3. Testing

Testing was carried out following exactly same process as validation. Best performing model was loaded and test on the provided test data. No data transformation was applied in the test dataset except the Channel-wise normalization.

Since the original format of the annotations was altered while training, so some post processing upon model predictions was required to convert them to the original format, before saving the testing phase prediction to CSV. Prediction for "upcolor" and "downcolor" were in form of integers which were converted to the strings holding name of the colors as shown in the figure 4 and 5 below.

	file_name	age	backpack	bag	handbag	clothes	down	up	hair	hat	gender	upcolor	downcolor
0	010014.jpg	1	0	0	0	1	1	1	0	0	0	6	3

Figure 5

	file_name	age	backpack	bag	handbag	clothes	down	up	hair	hat	gender	upcolor	downcolor
0	010014.jpg	1	0	0	0	1	1	1	0	0	0	upred	downgray

Figure 6

After that, categorical values were converted into dummy indicators; multiple columns for "upcolor" and "downcolor" values instead of only one column.

"downyellow" and "downpurple" columns were missing in predictions. May be because of that there was no image with "downyellow" and "downpurple" color. So were added manually. During pre-processing phase we modified data range, instead of starting from 1 we have modified to start it from 0, for each column. So now we resotred the original format as provided in training annotation data eg. now "age" will have range => [1, 2, 3, 4] instead of [0, 1, 2, 3]. After above post-processing, final predictions were saved to a CSV file.

Similarly for the second task of person re-identification, query images were passed to the model. Most similar images against each query image, predicted by the model, were saved to the out format requested for this task. For this part, one of the challenging phase was where we had to decide the number of images to retrieve per query. To make this decision we explored the statistical properties of the training set, we extracted data about the maximum number of images for each identity present in the training set and also the mean. The maximum was 72 and the mean was 17.3. After taking in consider the mAP evaluation metric, we decided to retrieve 100 images per query, such we concede some extra room for false negative.

We had tried also the elbow method, fixed a similarity threshold (0.85). If the similarity between the query image and the gallery image is equal or greater than the threshold, then consider the gallery image as a similar image of the query, otherwise discard. But the mAP score for this approach was very low.

2.4. Evaluation

For the evaluation of the classifier, **CrossEntropyLoss()** was used as cost function, it combines LogSoftmax() and NLLLoss() in one single class. So we just had to send raw output without applying LogSoftMax.
The model with the minimum validation loss was selected as the best performing model for the first task of image classification. We also monitored the accuracy of every attribute prediction. At the end, our best model had produced ***0.404*** as loss and ***0.86*** as average accuracy during the validation.

For the re-identification task, Mean average precision (mAP) was used as performance evaluation matric. Initially cosine similarity was calculated between the feature vectors extracted by the model for query images and gallery images which did not yield satisfactory mAP score. While after the implementation of a custom loss formulate, made y joining triplet loss and classification loss, model was able to perform good enough on the validation data and produced a mAP score of ***85.39*** **.** The following table shows some mAP score comparison between our three implemented models and the state of art approaches:

Methods	Publish	Backbone	mAP
GAN	ICCV-2017	Res50	66.07
APR	CVPR-2019	Res50	66.89
TriNet	CVPR-2017	Res50	69.14
ABD-Net	ICCV2019	Res50	88.28
Unsupervised Pre-training	CVPR-2021	Res101 + MGN	92.00
Classification model (our)		**Res50**	**65.61**
Triplet loss (our)		**Res50**	**71.84**
Joint loss (our)		**Res50**	**85.39**

Table 1: Comparison with state of the art on Market-1501

References

[1] O'Mahony, N.; Campbell, S.; Carvalho, A.; Harapanahalli, S.; Hernandez, G.V.; Krpalkova, L.; Riordan, D.; Walsh, J. Deep learning vs. Traditional Computer Vision. In Proceedings of the Science and Information Conference, Tokyo, Japan, 16–19 March 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 128–144. [Google Scholar].

[2] Voulodimos A, Doulamis N, Doulamis A, Protopapadakis E (2018) Deep Learning for Computer Vision: A Brief Review. Comput Intell Neurosci 2018:1–13. https://doi.org/10.1155/2018/7068349

[3] Nash W, Drummond T, Birbilis N (2018) A Review of Deep Learning in the Study of Materials Degradation. npj Mater Degrad 2:37. https://doi.org/10.1038/s41529-018- 0058-x

[4] CS231n Convolutional Neural Networks for Visual Recognition. http://cs231n.github.io/transfer-learning/. Accessed 9 June 2021

[5] Khandenwal, R., 2019. Deep Learning using Transfer Learning. [online] Medium. Available at: <https://towardsdatascience.com/deep-learning-using-transfer-learning-cfbce1578659> [Accessed 17 July 2021].

[6] "User guide: contents — scikit-learn 0.24.2 documentation", Scikit-learn.org, 2021. [Online]. Available: https://scikit-learn.org/stable/user_guide.html. [Accessed: 18- Jul- 2021].

[7] "ImageNet - torchbench Docs", Paperswithcode.github.io, 2021. [Online]. Available: https://paperswithcode.github.io/torchbench/imagenet/#input_transform. [Accessed: 18- Jul- 2021].

[8] D. Goswami, "Introduction to Early Stopping: an effective tool to regularize neural nets", Medium, 2020. [Online]. Available: https://towardsdatascience.com/early-stopping-a-cool-strategy-to-regularize-neural-networks-bfdeca6d722e. [Accessed: 20- Jul- 2021].

[9] R. Jie Tan, "Breaking down Mean Average Precision (mAP)", Medium, 2019. [Online]. Available: https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52. [Accessed: 21- Jul- 2021].