Person Re-Identification: CSCE 625-Artifical Intelligence Project Report

Akshaye Anitha Pradeep
akshayeap@tamu.edu

Rohan Agrawal
rohan@tamu.edu

Kevin McNeff
kevinmcneff@tamu.edu

Abstract

Person re-identification in the wild involves finding targeted people across different space-time locations, where occlusion and body postures pose significant challenges. Our problem statement involves matching specified images against a gallery of persons in another camera view by measuring the similarity between the images. We propose to use Attention mechanism in our Neural Networks to extract more discriminative information from images. We build a residual attention network, which can incorporate with state-of-art feed forward network architecture in an end-to-end training fashion, that can detect features that stand out which can then be used to classify the image. Additionally, we were planning to use Refined Part Pooling to match outliers incurred during the attention based partitioning and assigns them to the parts they most closely match, thereby enhancing consistency within the parts they are assigned to. We initially augment our input using random erasing so as to prevent over-fitting and to make our model more robust.

1. Introduction

Cross camera person re-identification has many uses that are impactful in the real world. As technology continues to become more prevalent in daily life, security and closed circuit cameras have become common within households, businesses and across cities. There exists a lot of data to train on, and any number of people or businesses using multiple surveillance cameras could find use in automatic person re-identification. This is one of the reasons why it is important to create a generalized and robust model for the problem. To build this generalized model, the model needs to be robust to varying image quality, size, crops, occlusion, scenes, and attentive to unique characteristics of a person to be identified. Both the Market1501 [15] and the DukeMTMC [9] datasets due to their size and to capture a variety of scenes and crop ratios. The three proposed methodologies have been chosen for their ability to answer this problem and create the sought generalized model.
We build a Residual Attention Network that generates attention-aware features by stacking attention modules. The features from different modules adaptively as the layers get deeper. The feedforward and feedback attention process is unfurled into a single feedforward process by the bottom-up top-down feedforward structure in each attention module. The high number of attention modules leads to capturing different types of attention and increases performance greatly. Distinctive types of attentions for the hot air balloon in Fig.1 demonstrates this. The balloon instance mask highlights the bottom portion of the balloon while the sky attention mask reduces background responses. Additionally, the network which can incorporate state-of-art feed forward network architecture in an end-to-end training, can be easily extended to a depth of hundreds of layers. All of the previously mentioned properties, which are challenging to attain with past approaches, are made conceivable due to a ***Stacked network structure***, by which different Attention Modules can capture different attention types, ***Residual Attention Learning***, which optimizes deep Residual Attention Network with hundreds of layers to reduce the performance drop caused due to stacking Attention Modules directly, and ***bottom-up top-down feedforward attention*** which allows us to create a network with top-down attention by imitating bottom-up fast feedforward process and top-down attention feedback into a single feedforward process. [11].

2. Previous Works

An attention-based model for recognizing multiple objects in images is proposed in [1]. In [14], the authors apply attention mechanisms to the problem of generating image descriptions. Application of visual attention to fine grained classification tasks using deep learning is discussed in [13]. Residual attention networks was proposed by [11] and their idea of building the network by stacking Attention Modules to generate attention-aware features was derived by us. Their Residual Attention Network achieves state-of-the-art object recognition performance on benchmark datasets including ImageNet. Market1501 [15] is a dataset and baseline implementing person re-identification. DukeMTMC [9] is another dataset designed for person re-identification. Both are used to form a dataset to improve robustness in combination with random erasing. [16] In [10], the authors

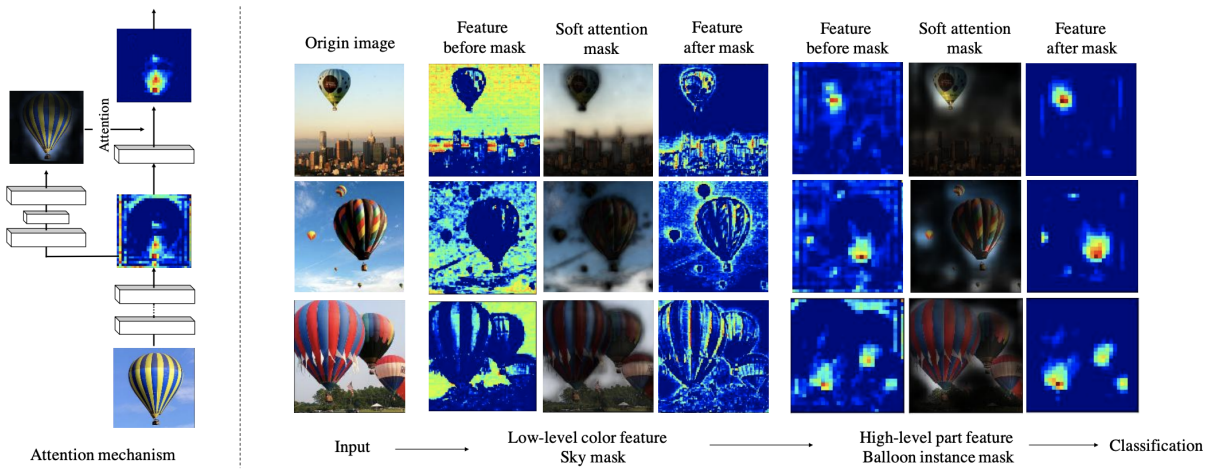


Figure 1. **Left.** The interaction between features and attention masks. **Right.** Examples illustrating different attention masks corresponding to different features. The balloon instance mask highlights the bottom portion of the balloon while the sky attention mask reduces background responses.

Figure 2. Examples of Random Erasing. [16]

correct the within-part inconsistency with the help of Refined Part Pooling using a part classifier.

3. Approach

3.1. Random Erasing

Random erasing is a data augmentation technique that fulfills several purposes. It can be effective in bolstering the training data size in small data sets. In our implementation, we used random erasing to make our model more robust to the generalized problem of person re-identification rather than simply the data set it was trained on. Random erasing takes random rectangles out images in the input data set and replaces them with pixels with random values. This method has been shown to reduce the risk of over fitting and becoming dependant on the training data. [16] The inclusion of random erasing increases robustness to occlusion. This means that if in testing or validation there are images where the persons to be re-identified are obscured by an object in front of them, training with random erasing will increase recall.

3.2. Residual Attention Network

Attention mechanism are based on the visual attention mechanism found in humans. It essentially means being able to focus on certain regions of an image, by viewing that region in higher resolution while making the surrounding objects lower in resolution. We stack attention models to create our attention network, which has two branches: trunk branch and mask branch. Feature processing is performed by the trunk branch. The output from the trunk branch, $T(x)$ with input x, is used by the mask structure in a bottom-up top-down structure that mimics the fast feedforward and feedback attention process [6, 8] to learn a mask $M(x)$ of the same size that produces soft weights of output features of $T(x)$. The output mask is used as control gates for neurons of trunk branch and the output of the attention module H is:

$$H_{i,c}(x) = M_(i,c)(x) \times T_{i,c}(x) \quad (1)$$

where i covers all spatial positions and c $\in \{1, ..., C\}$ corresponds to the index of the channel. The total structure can be trained completely.
For input feature in the soft mask branch, the gradient of

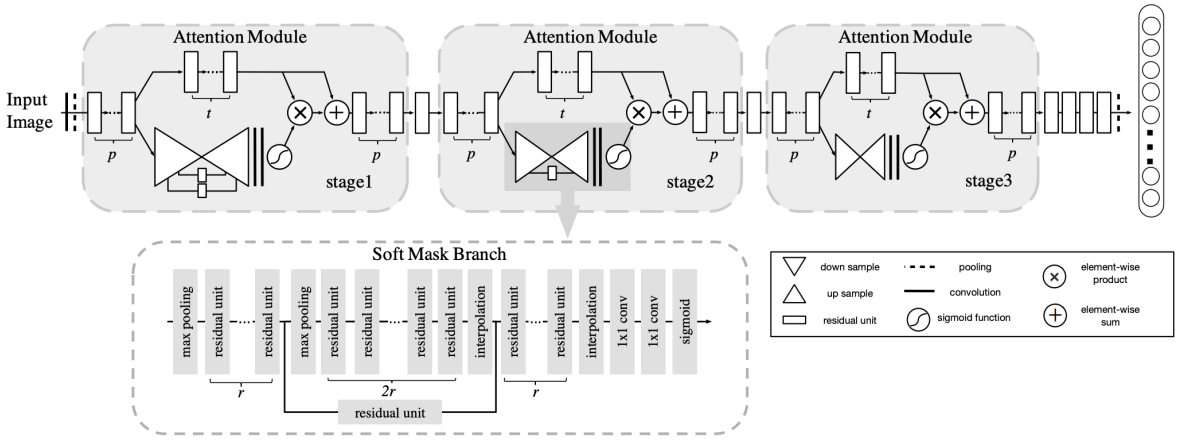


Figure 3. Example architecture of the proposed network. Three hyper-parameters p (the number of preprocessing Residual Units before splitting into trunk branch and mask branch), t (s the number of Residual Units in trunk branch) and r (the number of Residual Units between adjacent pooling layers in the mask branch) are used in the the design of Attention Module.

mask is:

$$\frac{\partial M(x,\theta)T(x,\phi)}{\partial \phi} = M(x,\theta)\frac{T(x,\phi)}{\partial \phi} \tag{2}$$

where the mask branch parameters and the trunk branch parameters are represented by θ and ϕ respectively. This property prevents attention modules from being affected by noisy labels and can prevent wrong gradients produced by noisy labels from updating trunk parameters.

We could have used a single network branch in our implementation, instead of stacking attention modules, but the former method has many drawbacks: Images with difficult scenes and background clutter, should be modeled by diverse sets of attentions. Employing a single mask branch to capture all combinations of diverse factors will take a huge number of channels . Also, having just the one Attention Module does not allow a second opportunity to modify features in case the alteration fails on a few areas of the image. With a Residual Attention Network, the stated issues are taken care of by having every trunk branch have its own mask branch to learn attention that is trained for its features. Additionally, attention for cluttered images can be refined due to the the additive behaviour of stacked network structure.

3.2.1 Attention Residual Learning

We use attention residual learning(ARL) to prevent degradation of feature values in deep layers and to prevent soft mask from potentially breaking good properties of trunk branch. We adjust output H of Attention Module as

$$H_{i,c}(x) = (1 + M + i, c(x)) \times F_{i,c}(x) \tag{3}$$

$M(x)$ takes values in the range $[0, 1]$. If $M(x)$ takes value 1, then $H(x)$ will approximate original features $F(x)$. This method is called attention residual learning.

Stacked ARL that we use in our model is distinct from residual learning. For example, in ResNet, residual learning takes the form

$$H_{i,c}(x) = x + F_{i,c}(x) \tag{4}$$

where the residual function is approximated by $F_{i,c}(x)$, whereas in our model, $F_{i,c}(x)$ shows the features generated by deep convolutional networks. This is ensured by our mask branches $M(x)$ which work as feature selectors to enhance good features.

3.2.2 Mask Branch

As mentioned before, our mask branch contains fast feedforward sweep, which quickly collects global information of the full image, and top-down feedback steps, which combines global information with original feature maps. Max pooling is executed a few times on the input to increment the receptive field quickly after a number of Residual Units. Once the least resolution has been reached, we then expand the global knowledge by a balanced topdown architecture to direct input features in every location. We then up sample the output by Linear Interpolation. To keep the output range equal to the size of input feature map, we make the count of bilinear interpolation the same as the count of max pooling. A sigmoid layer is then used to normalizes the output range to $[0, 1]$ after two consecutive 1×1 convolution layers. Skip connections between bottom-up and top-down parts are also added to obtain feature details from different scales[11] .

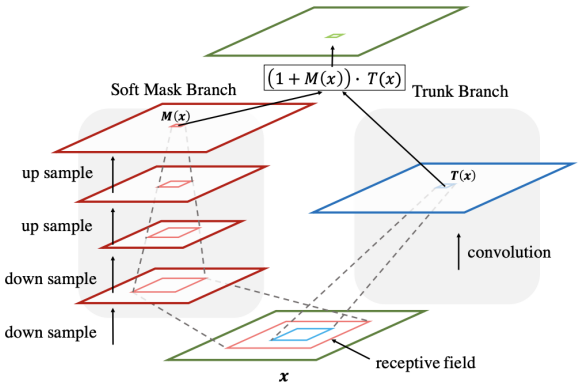


Figure 4. The receptive field comparison between mask branch and trunk branch.

3.2.3 Mixed, Spatial and Channel Attentions

Three types of activation functions are used by us, which correspond to mixed attention, channel attention and spatial attention[11]. Mixed attention f_1 use sigmoid for each channel and spatial location.

$$f_1(x_{i,c}) = \frac{1}{1 + exp(-x_{i,c})} \tag{5}$$

Channel attention, f_2, performs $L2$ normalization for every spatial position within all channels.

$$f_2(x_{i,c}) = \frac{x_{i,c}}{||x_i||} \tag{6}$$

Spatial attention f_3 performs normalization from each channel within feature map and then sigmoid to get soft mask related to spatial information only.

$$f_3(x_{i,c}) = \frac{1}{1 + exp(-(x_{i,c} - mean_c)/std_c)} \tag{7}$$

Where i ranges over all spatial positions and c ranges over all channels. $mean_c$ and std_c denotes the mean value and standard deviation of feature map from c-th channel. x_i denotes the feature vector at the i-th spatial position.

3.3. Refined Part Pooling

When an image is processed from all the layers of a backbone network, it transforms to a 3D tensor of activations. Column vectors f of this tensor should be similar in its group and dissimilar to other groups. If its not, then within-part inconsistency occurs.
Refined Part Pooling(RPP) is done to correct this within-part inconsistency. The attention mechanism also works based on the structure of Part-based Convolutional Baseline. With the use of RPP (w/o induction) setting, via attention, the network learns to focus on several parts , and achieves significant improvement over IDE, which learns a global descriptor.
We want to relocate the outliers among the column vectors based on their similarity within parts. We recursively sample the column vector f into part P_i according to the similarity value $S(f \leftrightarrow P_i)$, formulated as:

$$P_i = \{S(f \leftrightarrow P_i)f, \forall f \in F\} \tag{8}$$

where F is the complete set of column vectors in tensor , $\{\cdot\}$ denotes the sampling operation to form an aggregate. It is significant to measure the similarity between a given column vector f and every part. We need to repeat the process of measuring similarity followed by sampling till it converges because the similarities calculated previously are invalid now. This results in a significant clustering embedded in deep learning.

Refined Part Pooling uses a part classifier to predict the value of $S(f \leftrightarrow P_i)$ (the probability of f belonging to P_i), instead of computing similarities between the column vector f and part P_i, as mentioned in [10]. By doing this, the proposed refined part pooling refines the original hard and uniform partition by conducting soft and adaptive partition. The outliers resulting from the uniform partition will be relocated.

For learning the part classifier without labelling information, we use the comparison mentioned in [3] that compares RPP with another potential method derived from Mid-Level Element Mining(MLEM).Though MLEM can improve PCB but it is inferior to RPP. As mentioned, since no part labels are present, the part classifier of RPP and the ID classifier are jointly optimized to recognize training identities, and thus gains better pedestrian discriminative ability over Mid-Level Element Mining.[3] Due to time constraints, we couldn't implement this part to our project.

3.4. Cross Entropy Loss

A loss function in neural network is a way to measure how closely our model represents our dataset. If we are way off, it produces a higher value. So, the aim of a neural network is to minimize the loss function as much as possible. There are many kinds of loss function for example: Mean Squared Error, Log Loss (Cross Entropy Loss), Likelihood Loss. A proper choice of loss function is very critical for convergence of neural network. We have chosen Cross Entropy Loss as suggested in the baseline.

The cross entropy loss determines the loss between predicted and true probabilities. Given an image in our dataset, we know that it belongs to a particular person with probability 1 and to other persons with probability 0. That represents its true probabilities. Now the trained classifier also predicts probabilities based on its similarity to each person.

So the higher the probability it assigns the more accurate it is. Also the cross entropy loss will be lower.
Mathematically, let the true probability P_i is the true label, and the given distribution Q_i is the predicted value of the current model. Then true entropy is represented by

$$H(p) = -\Sigma_i p_i * log_2(p_i) \tag{9}$$

It represents the useful information present in the message. The cross entropy loss is represented by:

$$H(p, q) = -\Sigma_i p_i * log_2(q_i) \tag{10}$$

In code, variable 'labels' represent the ground truth and the variable 'output' represents the predicted probabilities. Then we use back propagation on this cross entropy loss to refine our weights.

3.5. Backpropagation

Backpropagation is a method of calculating the gradients of a artificial neural network to change the weights of a neural network that can be used in each new epoch. In supervised learning algorithms, we try to find functions that can best represent the mapping from our inputs to desired outputs. Backpropagation trains a multi-layered neural network to learn internal features so it can generate outputs from any arbitrary input. [12]

Gradient descent with backpropagation doesn't always ensure global minimum. It might get stuck in the local minimum since it might not be able to cross the plateaus of error function. The main reason behind it is the non-convexity of the error functions in neural networks.

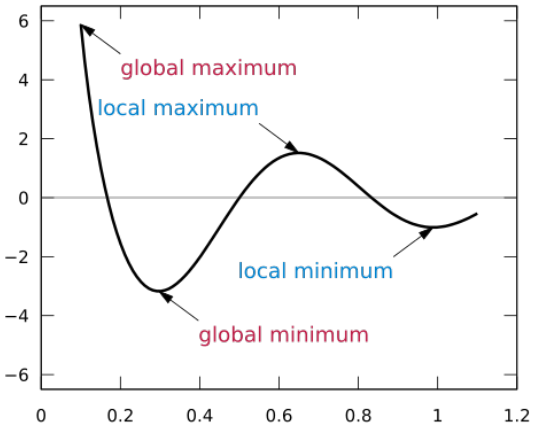


Figure 3.5: Examples of Non Convex Error Function. [12]

Normalization of input vectors can improve the back-propagation performance."[12]

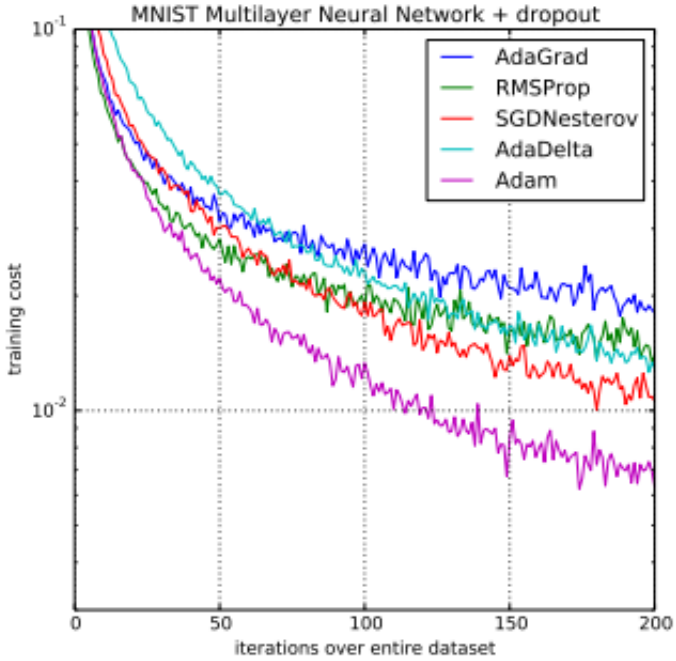
3.6. Adam Optimization

Adam is an efficient method with much lesser memory requirements, for stochastic optimization that only uses first-order gradients. It combines the power of two popular methods AdaGrad and RMSProp.[4]

AdaGrad: It stands for Adaptive Gradient Algorithm. It maintains a "per-parameter learning rate" that works well with sparse gradients problems ex. Computer Vision and Natural Language Processing.

RMSProp: It stands for Root Mean Square Propagation. It also maintains "per-parameter learning rates" that adapts itself based on the average of the recent magnitudes Adam utilizes the average of second moments of gradients as opposed to first moments as in RMSProp.

Comparison of Adam to Other Optimization Algorithms Training a Multilayer Perceptron can be visualized in the figure below:[Source:[2]]



As summarized in [2], Adam offers following **advantages** over non-raised advancement issues:
1. Little memory necessities.
2. Computationally effective.
3. Easy to implement.
4. Invariant to diagonal rescale of the gradients (Diagonal rescale involves multiplying the gradient by a diagonal matrix with only positive factors).
5. Appropriate for problems like Person Re-Identification that are demanding in terms of data as well as parameters.
6. Good for non-stationary objects. So, it would be suitable

in our case since it involves non stationary pedestrians generally.
7. Hyper-parameters require less tuning.

Hence, Adam was chosen as the optimizer in this project.

3.7. Step Learning Rate

Learning rate determines how quickly we are trying to minimize the loss function i.e. we are trying to find the global minimum of our function. If its too steep, it overshoots the minimum and takes more time than expected. If its too low then it takes a significant amount of time for the network to converge. So to optimize it, we are using a learning rate schedule that changes the learning rate as training progresses further.

For example, the step decay learning rate and its impact on accuracy can be visualized from the two figures below [Source:[5]]:

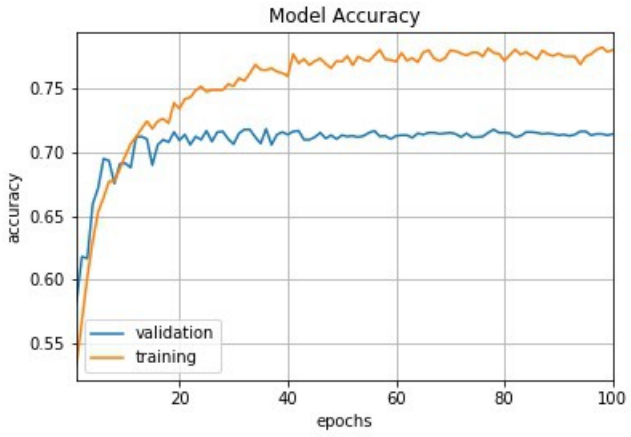


Figure 3.7: Step Decay Schedule

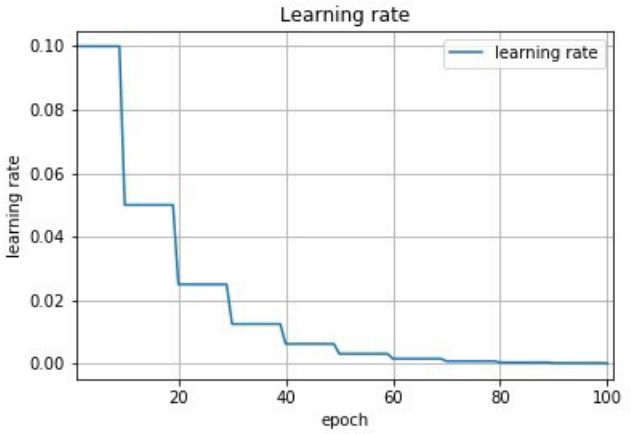


Figure 3.7: Step Decay Schedule

We deployed a step based decay for our learning rate. After every 40 epochs, we are reducing the learning rate by a factor of 0.1. So as we tend to reach lower values of loss, it decreases the learning rate and searches slowly for the minimum.
This combined with Adam optimizer, helps to "reduce the loss during latest step of training, when the computed loss with the previously associated maximum update has stopped to decrease." [Source:[7]]

4. Training

For training, we used 8 gb Nvidia GPU. We ran 25 epochs and with 2 number of workers(threads). It took around 12 hours to complete the training. For training, we used Attention mechanism and Refined Part Pooling. For loss function, we used cross entropy loss function along with back-propagation. For optimizer, we used Adam Optimizer. We decayed our learning rate by a factor of 0.1 for every 40 epochs.
We chose a combination of Market-1501 and DukeMT for training our model. These datasets have around 32668 and 36411 images respectively and 1501 and 1812 identifiers. Other datasets like MSMT and CUHK03 were skipped due to hardware constraints and small size respectively.

4.1. Epoch Statistics

We calculate running loss and running corrects normalized by dataset size for each epoch for each phase training and validation respectively. We use this loss and the error i.e. $(1 - accuracy)$ and plot it in two separate graphs for each epoch iteration. This helps to visualize the performance of the system. Also, after every 10th epoch, we are dumping the network parameters to .pth file with name net_epoch_label. The last dumped file is used to test the new dataset.

5. Testing

While testing, we are flipping the given set of images to augment detection. The following parameters are evaluated as part of testing:

5.1. top-1 score

Whether the top class (the one which the highest probability) is the same as the target label

5.2. top-5 score

Whether the target label is one of the top 5 predictions (the 5 ones with the highest probabilities).

5.3. mAP (mean Average Precision)

The mAP can be calculated from:
Precision = True Pos / (True Pos + False Pos)
Recall = True Pos / (True Pos + False Neg)

AP (average precision) =average of maximum precision at these 11 recall levels

6. Results

We achieved the following results on the given testing set:

1. top1: 0.773
2. top5: 0.9
3. top10: 0.9533
4. mAP: 0.6175

7. Conclusion for Future Work

Based on this project and other teams' efforts, we can conclude that the following aspects can be crucial to make our neural network converge faster and accurately:

1. Dataset size for training:
The bigger the better. Teams that chose a number of diverse datasets performed better than teams who only used one or a couple small datasets. If the time and hardware is there, then there is little incentive to use less data as long as the standard for the datasets is held high.

2. Loss function:
The right choice is specific to the chosen algorithm and outliers in the dataset.

3. The learning rate:
Learning rate plays a significant role in converging the network. Adaptive learning rates are better than the static ones.

4. Number of epochs:
Epochs represent the number of iterations for our neural network. Less epochs produce very high error rates while too many epochs may stop showing any significant improvement once the global minimum is near or the network is unstable, probably due to high learning rate. So, it must be optimized while running the model.

5. Number of threads based on hardware limitations.
Multiprocessing must be used keeping hardware limitations in mind.

8. Contribution

All the members contributed equally in various aspects ranging from coding, design, logistics and theory.

References

[1] J. Ba, V. Mnih, and K. Kavukcuoglu. Multiple object recognition with visual attention, 2014.

[2] J. Brownlee. Gentle introduction to the adam optimization algorithm for deep learning.

[3] V. Ferrari. *Computer Vision – ECCV 2018*.

[4] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[5] S. Lau.

[6] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, June 2015.

[7] nessuno.

[8] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. *Lecture Notes in Computer Science*, page 483499, 2016.

[9] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.

[10] Y. Sun, L. Zheng, Y. Yang, Q. Tian, and S. Wang. Beyond part models: Person retrieval with refined part pooling. *CoRR*, abs/1711.09349, 2017.

[11] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, and X. Tang. Residual attention network for image classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[12] Wikipedia contributors. Backpropagation — Wikipedia, the free encyclopedia, 2018. [Online; accessed 9-December-2018].

[13] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, and Z. Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 842–850, 2015.

[14] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention, 2015.

[15] L. Zheng, L. Shen, L. Tian, S. Wang, J. Wang, and Q. Tian. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*, 2015.

[16] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.