Amur Tiger Re-Identification

Sainath Reddy Bobbala, Sneha Oladhri

University of South Florida

Abstract

We propose a re-identification network for Amur tigers [11] using convolution neural networks. Re-identification is a widely used use case across many computer vision problems like object tracking, person surveillance. The main goal in our approach is to identify the difference in stripes of the tigers as each tiger have different stripes as in Fig.1 similar to humans having different thumbprint. Initially, we propose a baseline method for the re-identification of tigers using the pre-trained Resnet-50 model which was trained on the Imagenet dataset. Further, we introduce the part-based model to handle the pose variations in tigers. We evaluate the model using an mAP metric.The source code is available at https://github.com/usfproject/amur_tiger/tree/master.

Figure 1. every tiger will have unique stripes similar to humans having different patterns on thumbs

1. Introduction

Monitoring wildlife is very important and especially in the case of endangered species like tigers. Amur tigers were one of the largest and most endangered species in the world. These animals are mostly threatened by habitat loss, poaching of their prey, and some infectious diseases. Traditional tagging methods do not scale to large populations so using computer vision-based re-identification makes this problem interesting and scalable to large populations. It helps in monitoring the movement of an individual animal, keeping track of the population of a species, controlling animal theft, and hence helps in safeguarding endangered species. And this Re-identification of animal images in wild is a challenging task as there may be subtle differences between the animals, the difference may be in body marking or shape. Person re-identification is a similar task. But re-identifying tigers are difficult to compare to people as people will have additional artifacts like clothes, bags, etc which aid in re-identification. In re-identification using deep learning the object that needs to be re-identified is represented using embedding. Embedding is the low-dimensional, learned (through CNN training) continuous vector representations of the object, in our case each tiger will be represented with embedding. And this embedding will be used to find the tiger if it is in a different scene. In our implementation, we will mostly follow the approach given in [12].

2. Background

2.1. Person Re-identification

Person Re-identification is a well-defined problem using deep learning. Using deep learning, generally, people aim at developing effective CNN's using different losses, like classification loss, triplet loss, etc. In spatial-temporal person re-identification[16], they use two streams for modeling. One for visual stream and the other for spatial-temporal stream. Few implementations also consider the pose of a person for re-identification.

2.2. Animal Re-identification

Similarity network was developed in [15] i.e., the Siamese network for finding the similarity between the animals. Animals with high similarity were given the same ID. In the following paper[13], strong baseline tricks for getting better results on deep person re-identification have been mentioned. Warm-up Learning rate, random erasing augmentation, Batch normalization, triplet loss, center loss were few concepts discussed in the paper. In the following implementation[4], feature extraction was implemented by uniformly partitioning the feature maps and also by using pose information provided in the dataset to divide feature maps.

3. Data-set

In our proposal, we use the Amur tiger dataset provided in CVWC 2019 challenge[1]. The dataset consists of 92 tigers from 10 zoos. Data-set contains cropped images with manual annotated ID and key-points. As stripe information is critical for tiger re-identification, the left and right sides of the tiger body were stored as two entities for a tiger. Sample images of the tiger :

Figure 2. Example 1

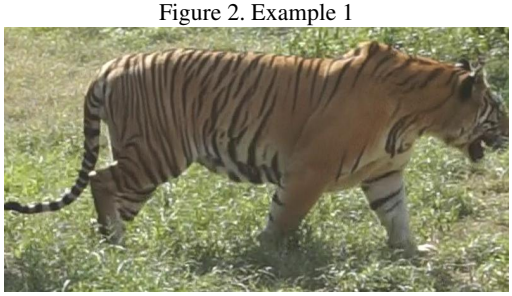

Figure 3. Example 2

The dataset also contains the keypoint descriptions of the tiger. These key points are used for extracting parts in the PPGNet model. A total of 15 feature key-points was provided which are mentioned in the below table. The annotations were provided in the coco format for key-points.

keypoint	Keypoint name	keypoint	Keypoint name
1	center point	9	right knee
2	nose	10	left knee
3	right front paw	11	left back paw
4	left front paw	12	right back paw
5	root of the tail	13	right shoulder
6	right ear	14	left shoulder
7	left ear	15	left hip
8	right hip		

Table 1. Keypoint features

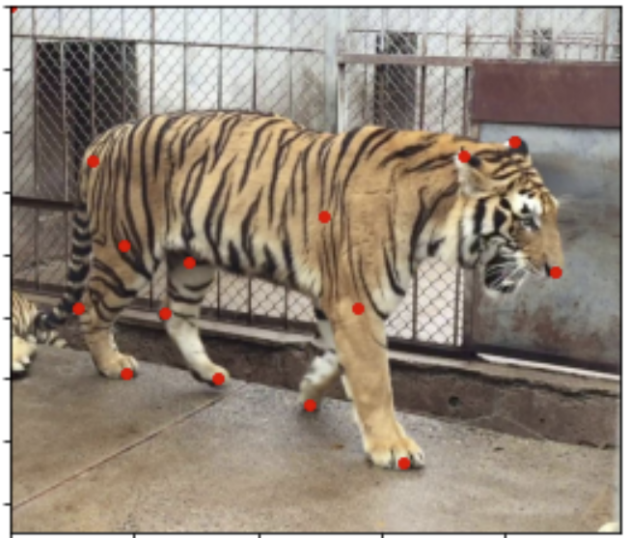

Figure 4. Keypoints plotted in the tiger

The dataset is divided into train and test. Train dataset contains 1887 images and test dataset contains 1765 images.

Figure 5. Data Distribution

There are total 107 entities in the dataset. Entity distribution in the dataset can be viewed below:

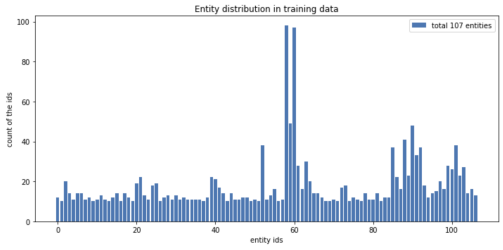

Figure 6. Entity Distribution

4. Proposed Method

4.1. Baseline

Here we will discuss two baseline models that we have re-created as discussed in [11]. First, we will discuss the

classification model later followed by using the triple- loss based trained model as a baseline.

4.1.1 Classification based baseline

Models trained on the Image-net dataset are been widely used as feature extractors, we have used Resnet-50 [8] which has 50 resnet blocks trained on Imagenet. Resnet-50 acts as a backbone network on top of it we have added 2 more layers of fully connected (FC) layers of size 1024 and 107 layers where 107 is the number of tiger entities in the train set. While training to avoid overfitting layers of Backbone layer i.e Resnet 50 is frozen and error residuals are backpropagated to only last two FC layers, and also image augmentation are added randomly and will discuss in further sections.

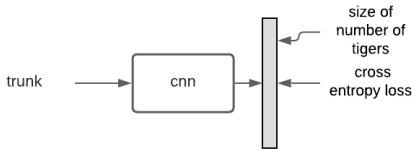

Figure 7. baseline model

4.1.2 Triplet loss based baseline

For the tasks of person and wildlife re-identification, both the cross-entropy loss and the triplet loss [9] equation 2. have been widely used for optimizing the network. For the same network discussed in section 4.1.1, we remove the last FC layer i.e output classification layer and train it with triplet loss keeping the same network settings.

$$\mathcal{L}(A,P,N) = \max\left(\|\,\mathrm{f}(A)-\mathrm{f}(P)\,\|^2 - \|\,\mathrm{f}(A)-\mathrm{f}(N)\,\|^2 + \alpha, 0\right) \quad (1)$$

There are various forms of triplet loss based on how we choose Anchor, Positive, Negative image pairs. Anchor, Positive is the images of the same tiger entity, whereas the negative image is of a different tiger entity. If we choose a Positive and Negative image randomly for a given Anchor image there will be three situations. Easy triplet - where Positive image is very similar to Anchor and Negative image is very different from Anchor. This is shown in equation number 2 where α is a margin value.

$$\|\,\mathrm{f}(A)-\mathrm{f}(P)\,\|^2 + \alpha < \|\,\mathrm{f}(A)-\mathrm{f}(N)\,\|^2 \quad (2)$$

Semi-Hard triplet - where Positive image is similar to Anchor and Negative image is different from Anchor. But the distance between positive and anchor plus margin is still less than the difference between an anchor and negative. This is shown in equation number 2 where α is a margin value.

$$\|\,\mathrm{f}(A)-\mathrm{f}(P)\,\|^2 < \|\,\mathrm{f}(A)-\mathrm{f}(N)\,\|^2 < \|\,\mathrm{f}(A)-\mathrm{f}(P)\,\|^2 + \alpha \quad (3)$$

Hard triplet - Negative tiger image is much closer to the anchor image than the positive tiger image, we have used hard triplet mining strategy while training.

$$\|\,\mathrm{f}(A)-\mathrm{f}(P)\,\|^2 > \|\,\mathrm{f}(A)-\mathrm{f}(N)\,\|^2 \quad (4)$$

we have followed an online triplet hard mining approach similar to open pose [5]. At the train time given a batch of N images, we intend to select the positive image which is far away from the anchor image, and the negative image is selected such that it is near to the anchor image.

$$\mathcal{L}(A,BP,BN) = \max(\max(\|\,\mathrm{f}(A)-\mathrm{f}(BP)\,\|^2) - \min(\|\,\mathrm{f}(A)-\mathrm{f}(BN)\,\|^2) + \alpha, 0)) \quad (5)$$

Here BP, BN represents a batch of positive and negative images. So for a given anchor image loss is calculated using equation 5.

4.2. Part-Pose Guided Network (PPGNet)

We will use this method mentioned in 3.1 as our baseline and then improve upon it by using the following approach. To do re-identification, representation of the data is very important so to accurately re-identify the tigers focusing on small features like textures on limbs, the body is very important. So instead of relying on single input networks which take the entire image to learn the embedding, we will study various other ways to divide the image so that local features which are very small in the image get more priority. One such method is used in [12]. where the networks have three streams full image stream (F), limb stream (LP), trunk stream (TP), Each stream is a separate CNN network. Full image stream takes the entire tiger image as input and uses an imagenet trained backbone network and gives a vector embedding. Limb stream takes each limb as input (limbs can be cropped from the pose annotations provided in the training data). And Trunk stream takes the body part of the network as the input and outputs a single vector. Then the full stream is combined with the trunk and limb stream separately.

within Limb stream (LP) identical parts are combined as part of tackling the spatial misalignment problem. spatial misalignment is been a challenging problem in the re-identification task due to large variations in pose, view angle, background clutter, illumination, etc as discussed in

[17]. so by combining identical parts across the network the spatial alignment factor is maintained.

We have initialized full image stream with Resnet-101 [8], but in the paper, they have initialized with Resnet 151, and Trunk stream and limb stream are initialized with Resnet-34. Resnet-101, Resnet-34 are pre-trained on Imagenet [14]. And from these pre-trained networks only bottleneck features are extracted and sent to the Batch Normalization layer [10] followed by a fully connected layer (FC) layer of 107 neurons i.e number of entities in the training set.

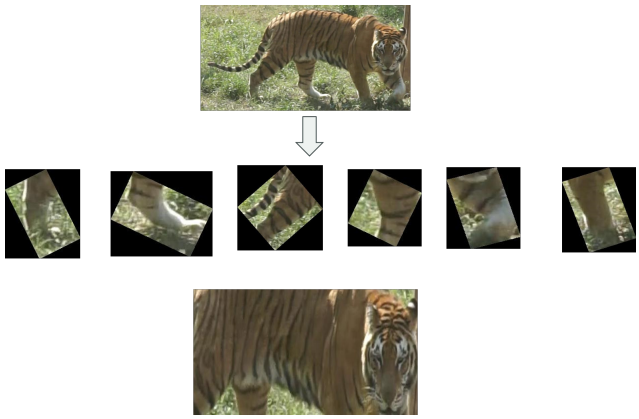

Figure 8. Extracted parts

4.2.1 Part images extraction

Parts were extracted from the tiger as mentioned in [12]. 7 parts from the tiger were extracted for Part Pose Network. Those parts include the trunk, front legs, hind thighs, and hind shanks. For extracting the trunk, a rectangle was drawn using the ears, nose, root of tail, shoulder key-points. Front legs are extracted using shoulder and front paw. Hind thighs are extracted using hips and knees key-points. Hind shanks are extracted using knees and back paws key-points. The width of the bounding box of hind shanks, thighs, front legs was set to 1/6,1,1/6 of its height respectively. Random noise was also added for robustness. Extracted front legs, hind thighs, and hind shanks were combined to the black background. Figure 10 shows the key-points plotted on the tiger and the bounding boxes of 7 parts.

4.3. Data augmentation

We used data augmentation techniques like normalizing the image, resizing the image to 256*128 (for baseline models), horizontal flip, rotation of image for 10 degrees, random-crop, padding, noise. So by using data augmentation on tigers with a fewer number of samples we will avoid the class imbalance problem and help in learning weak classes. And also adding more data using augmentation reduces model overfitting on train data. Before starting the training each image in the train set is flipped horizontally doubling the number of training samples.

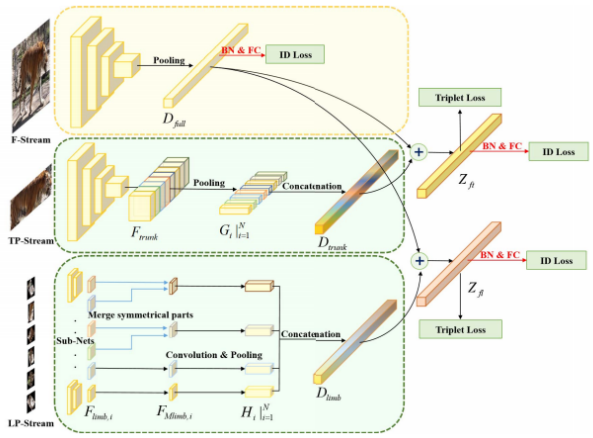

Figure 9. It consists of three streams: one F-Stream and two part streams. The F-stream is the main stream and used to extract the global feature from the full tiger image, while the two-part streams, the TP- and LP- streams, are used to extract local features from the trunk image blocks and limb image blocks, respectively. In the training stage, the local features play the role of regulator for the global feature learning. In the inference stage, only the F-stream is used for tiger re-ID. It consists of three streams: one F-Stream and two-part streams. The F-stream is the mainstream and used to extract the global feature from the full tiger image, while the two-part streams, the TP- and LP- streams, are used to extract local features from the trunk image blocks and limb image blocks, respectively. In the training stage, the local features play the role of regulator for the global feature learning. In the inference stage, only the F-stream is used for tiger re-ID. picture from [12]

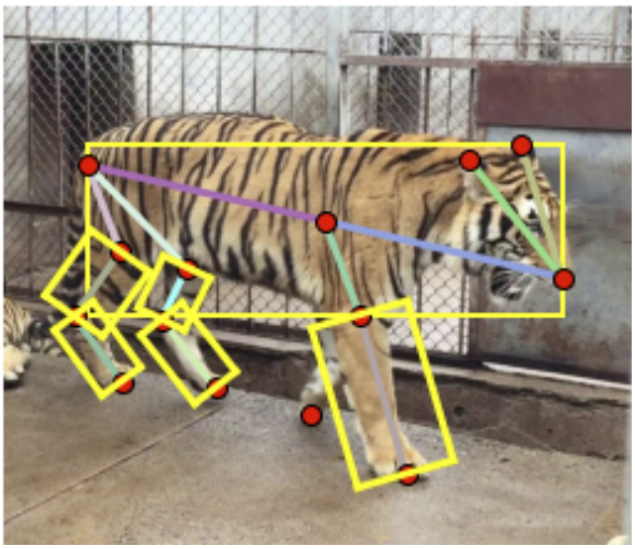

Figure 10. Bounding boxes on trunk, front legs, hind thighs and hind shanks

5. Evaluation

For evaluation, we use the mean average precision (mAP) and the rank-k accuracy as the metric to evaluate the algorithm. Each query image is divided into two fields: single camera, where the target tiger appears only in a single camera, and cross camera where the target appears in multiple cameras. The average of mAP on both the single-cam and the cross-cam case was used to evaluate the model.

6. Experiments

6.1. Baseline models

We performed experiments using different distances for distance matrix, with and without re-ranking of the labels for 150 epochs and batch-size of 535(number of unique class labels *5). Models were evaluated on two loss functions, Cross-entropy and Triplet. Initially, it was trained on 80-20 split data to find the epochs where validation was converging and then retrained for the same name of epochs on the whole training dataset. All the results were summarized in Table 1. We have used PyTorch to implement these networks and we have done the training in Google Colab.

6.2. PPGNet model

Compared to baseline approaches image size is increased here for full image stream image size is (256, 512,3) and trunk part image is resized to (64,128,3), and limb part image is resized to (64,64,3). The ID loss for the FStream, the ID loss, and triplet loss for the fused features (Fstream + LPstream, Fstream+ TPsream) are weighted by 1.0, 1.5, and 2.0 respectively.

The training is optimized by Adam optimizer using 500 epochs and with a batch size of 16 but in a paper batch size of 32 was used but with GPU size constraints we ended up using a lower batch size. When we first took the static learning rate then the model did not converge and heavy overfitting was observed. Then we adopt the warm-up strategy to bootstrap the network for better performance, where we spent 25 epochs linearly increasing the learning rate from $1 * 10^{-4}$ to $2.5 * 10^{-3}$. Then, the learning rate is decayed by a factor of 0.5 for every 80 (epochs is shown in the paper. This approach of changing learning rate is discussed in this paper [7]. 4 - fold cross-validation strategy is used for training. In our experiments, we have used the PyTorch framework for doing various training. For training one fold on Gaivi 12 GB GPU server, it took approximately 3 to 4 days for one fold. In the end distance matrix from 4 fold models are normalized and averaged for evaluation. The final results are summarized in Table 4. As shown in Table 4 we could not still achieve the results on par with the original implementation, This may be due to differences in implementation, In the paper, they have used Resnet-152 and batch size of 32 but due to computing restrictions, we used Resnet-101 with batch size 16. And also in the official implantation of PPGNet, they have used augmentation like Affine transformation, random erasing which may have aided in performance improvements in cross-cam scenarios.

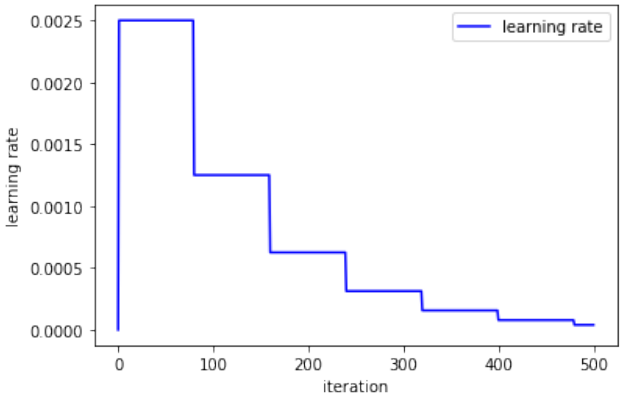

Figure 11. learning rate changing

6.3. Re-Ranking

Once we have the initial distance matrix, re-ranking has been used prove to improve mAP results as discussed in Re-ranking Person Re-identification with k-reciprocal Encoding [18]. Re-ranking has shown significant improvement in results as shown in Tables 1 and 2.

[H]

Parameter	Single-cam mAP	Cross cam mAP	Single cam top-5	Cross cam top-5
CE (ATRW)	59.1	**38.1**	**92.7**	**87.8**
CE+re-rank+cosine (Ours)	**62.0**	35.3	90.2	85.7
CE+cosine (Ours)	46.6	28.4	92.5	81.7
CE+re-rank+euclidean (Ours)	60.0	35.6	90.2	84
CE+euclidean (Ours)	50.0	29.4	91.1	84.0

Table 2. Results Table with CE loss

Parameter	Single cam mAP	Cross cam mAP	Single cam top-5	Cross cam top-5
triplet (ATRW)	71.3	**47.2**	**96.0**	**90.6**
triplet+ cosine (Ours)	56.6	32.7	**96.0**	89.1
triplet+ re-rank+ cosine (Ours)	**71.8**	39.3	**96.0**	89.7
triplet+ euclidean (Ours)	49.6	29.7	92.0	90.2
triplet+ re-rank+ euclidean (Ours)	67.6	38.4	94.0	89.7

Table 3. Results Table with triplet loss

Parameter	Single cam mAP	Cross cam mAP	Single cam top-5	Cross cam top-5
PPbm-b (ATRW)	72.8	47.8	95.6	90.7
PPGNet	**90.6**	**72.6**	**99.1**	**96.7**
PPGNet+ cosine (Ours)	72.7	44.1	98.2	89.1
PPGNet+ re-rank+ cosine (Ours)	83.2	49.4	98	90.2
PPGNet+ euclidean (Ours)	72.2	44.3	97.7	88.5
PPGNet+ re-rank+ euclidean (Ours)	83.0	49.4	98.2	88.5

Table 4. Results Table with PPGNet

Method	mAP single cam	top1 acc single cam	top5 acc single cam	mAP cross cam	top1 accuracy cross cam	top-5 acc cross cam
CE baseline	62.0	82	90.2	35.3	68	85.7
Triplet-loss baseline	71.8	88	96	39.3	**80**	**89.1**
PPGNet	**83.2**	**93**	**98**	**49.4**	**80**	90.2

Table 5. Results Summary

7. Code Reuse

Both the cross-entropy baseline and triplet loss baseline are implemented from scratch. Data loading, Data pre-processing, training, visualization, evaluation pipelines are written from scratch. Implementation mentioned in the repository [2] was referred for key-points parts extraction. We have reused a function in the re-ranking pipeline from the official implementation of [18]. For the online triplet loss hard mining approach we have reused the code from [3].

8. Conclusion and Future Work

In this work, we have explored and learned about Supervised tiger Re-Identification. we have learned how to train complex Pytorch models by writing custom functions. we have learned how to Build Part Pose Based model using pose data for re-identification of the tigers. we have explored different data augmentation approaches to match the characteristics of tigers in the validation and test set. we have also explored the Re-ranking method for re-identification of the tigers. As future work, we thought of implementing the query [6] expansion approach which has shown improvement in various re-identification tasks. And also we see that most of the ideas in tiger re-identification are borrowed from Person re-identification tasks, including PPGNet even though we have gone through various person re-identification resources, we still need to explore a lot of new techniques.

9. Timeline for weekly tasks and Division of Labor

Weekly timeline report:

Week 1
Data Exploration and visualization Preparing the dataset ready-to-run Implementing data augmentation pipeline
Week 2
Examining the ways to extract unique feature extraction from tigers like stripes Examining different preprocessing techniques which best suites the problem Building baseline model Mid-term Report
Week 3
Finish Building an end to end network for training Training the model to achieve better results
Week 4
Different other ways to improve the achieved results from the papers cited in 2.2, 2.3 and others

Figure 12. test image on the left and its top-5 matches on the right.

Week 5
Project Presentation Preparation Final Report Writeup

Task	Name
Exploratory Data Analysis	both
Data Visualization	sneha
Preparing Dataset to Run	sneha
Data Augmentation pipeline	sneha
Classification model	both
hyper parameter tuning and training	both
Triplet model	both
Parts Extraction	sneha
PPGNet pipeline	sainath
evaluation pipeline	sainath
re -ranking	sainath
results visualtion	sainath

Table 6. Division of Labor

References

[1] https://cvwc2019.github.io/challenge.html - amur tiger re identification challenge link. 2

[2] https://github.com/lcenarthas/cwcv2019-amur-tiger-re-id - ppgnet official implementation. 6

[3] https://github.com/negation/onlineminingtripletloss we have used online. 6

[4] https://github.com/vvfshadow/feature-aggregation. 1

[5] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. Openface: A general-purpose face recognition library with mobile applications, 2016. 3

[6] Dengpan Fu, Bo Xin, Jingdong Wang, Dongdong Chen, Jianmin Bao, Gang Hua, and Houqiang Li. Improving person re-identification with iterative impression aggregation. *IEEE Transactions on Image Processing*, 29:9559–9571, 2020. 6

[7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2018. 5

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 3, 4

[9] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network, 2018. 3

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015. 4

[11] Shuyuan Li, Jianguo Li, Hanlin Tang, Rui Qian, and Weiyao Lin. Atrw. *Proceedings of the 28th ACM International Conference on Multimedia*, Oct 2020. 1, 2

[12] Cen Liu, R. Zhang, and Lijun Guo. Part-pose guided amur tiger re-identification. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 315–322, 2019. 1, 3, 4

[13] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification, 2019. 1

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. 4

[15] Stefan Schneider, Graham W. Taylor, Stefan S. Linquist, and Stefan C. Kremer. Similarity learning networks for animal individual re-identification - beyond the capabilities of a human observer. *CoRR*, abs/1902.09324, 2019. 1

[16] Guangcong Wang, Jianhuang Lai, Peigen Huang, and Xiaohua Xie. Spatial-temporal person re-identification, 2018. 1

[17] Zhizheng Zhang, Cuiling Lan, Wenjun Zeng, and Zhibo Chen. Densely semantically aligned person re-identification, 2019. 4

[18] Zhun Zhong, Liang Zheng, Donglin Cao, and Shaozi Li. Re-ranking person re-identification with k-reciprocal encoding. *CoRR*, abs/1701.08398, 2017. 5, 6