CS230

Real-time Dog Breed Localization and Classification

Zinan Hu
Department of Mechanical Engineering
Stanford University
zinanhu@stanford.edu

Yulin Huang
Department of Mechanical Engineering
Stanford University
yuh245@stanford.edu

Genggeng Zhou
Department of Mechanical Engineering
Stanford University
g9zhou@stanford.edu

1 Introduction

As the applications of autonomous systems become broader, the urgency for object detection that enables higher-level automation increases. With the evolution of Deep Learning and the availability of large datasets, many researches have demonstrated the feasibility of training deep networks for object detection in images. Motivated by this development, we aim to apply and train a fine-grained neural network to distinguish between different breeds of dogs. This network will label dogs' locations in images using bounding boxes and identify their breeds. The ability for multi-object fine-grained detection in real-time can be adapted and generalized further for monitoring endangered animal species. While state-of-the-art deep learning models can achieve high accuracy and precision in object classification and localization, there have been some challenges in fine-grained image classification. In most computer vision tasks, information on image background may help identify a specific category. For example, if the background of an image is identified as the sky, we may assume the object in the image is related to something that can fly. However, in the case of fine-grained classification, such as dog breed identification, information on the background may be less valuable because target classes are more likely to appear in similar backgrounds. As a result, when performing fine-grained classification, it is necessary to introduce a mechanism to focus on important features in the image, such as different body parts of a dog in the case of dog breed identification.

2 Literature Review

This project has two main focuses, real-time object detection, and fine-grained classification. There is a vast amount of literature in both areas, but not so much on the intersection of the two. There are two main model structures in real-time object detection, YOLO and R-CNN. YOLOv4 (You Only Look Once) is a model with increased accuracy and speed for object detection that can be trained with limited computational powers compared to EfficientDet[1]. Building upon it, YOLOv5 reaches state-of-the-art performance[2]. Fast R-CNN is a structure that significantly increases training and detection speed while improving the accuracy, especially for sparse object proposals, which were an issue costly in time before[3]. Based on Fast R-CNN, Faster R-CNN introduces a Regional Proposal Network(RPN) as an attention mechanism and shares convolutional features between the networks to improve speed and accuracy[4]. There are also multiple approaches to fine-grained classification. Lin et al.proposed a Bilinear CNN that represents images as a dot product of features extracted from two CNNs to identify localized features[5]. Their model achieves high accuracy on various datasets and can process at high speed on a GPU. Xiao et al. [6] used a two-level attention model in a

Deep CNN by applying visual attention bottom-up, top-down, and part-level top-down to localize discriminative parts. Similarly, Zheng et al. [7] proposed a Multi-attention CNN that consists of convolution, channel grouping, and part classification sub-networks to generate parts from spatially correlated channels and classified based on those parts. At the intersection, Zhang et al. [8] modified YOLOv5 and integrated it with Coordinate Attention for positional information and Context Feature Enhancement Module (CFEM) for context information from multiple receptive fields to pay special attention to small objects.

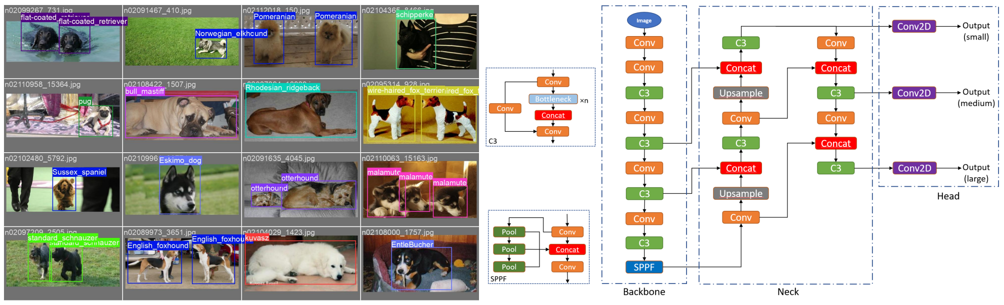
3 Dataset and Data Processing

3.1 Dataset

The dataset used for this project is the Stanford Dog Dataset [9]. It has 20,580 images of 120 breeds of dogs. The dataset is divided into 12,000 images in the training/validation set and 8,580 images in the test set. The dataset provides labels of dog breeds and corresponding bounding boxes in each image, saved in *xml* format. An example of a selected image from the dataset is shown in Figure 1a. All images in the dataset have a resolution of 640 pixels. One potential limitation of the dataset is that there are only 100 images for each breed of dog, and therefore may not be sufficient to train a large model.

3.2 Data Processing

The original dataset is not compatible with the YOLO model we are using. We developed scripts to preprocess the dataset. See details in Appendix A.



(a) (b)

Figure 1: (a) Examples from Stanford Dog Dataset with labels and bounding boxes. (b) YOLOv5 model structure

4 Baseline and Evaluation Metrics

4.1 Baseline Model

The general structure of YOLOv5 can be seen in Figure 1b. It consists of three major components: CSP-Darknet53 as the Backbone, SPPF (Spatial Pyramid Pooling - Fast) and CSP-Pan as the Neck, and YOLO Detection as the Head. The Backbone extracts features from the image at different granularities and passes those features to the Neck; the Neck then mixes and combines those features and passes them to the Head; the Head uses those features to make predictions about classes and bounding box positions for different anchors.

There are three fundamental blocks in the YOLOv5 model: CONV, C3, and SPPF. A CONV block consists of a 2D convolution with batch normalization and a Sigmoid Linear Unit (SiLU) activation function. Let the input be x, a CONV block can then be represented as $\text{CONV}(x) = \text{SiLU}(\text{BatchNormal}(\text{Conv2d}(x)))$. The C3 block consists of several CONV blocks, bottleneck blocks, and skip connections. An SPPF block performs consecutive maxpooling and concatenates the result of each maxpooling together. Their diagrams are illustrated in Figure 1b.

YOLOv5 has five different sizes, ranging from 1.9 million parameters to 86.7 million parameters. Due to limited time and computational resources, YOLOv5s and YOLOv5m are used as the baseline models. They have 7.2 million and 21.2 million parameters, respectively, which require reasonable training time and memory.

4.2 Evaluation Metrics

Several standard metrics are chosen to evaluate model performance: Precision, Recall, mAP50, and mAP50-95. Precision calculates the percentage of correct predictions; Recall calculates the percentage of classes identified correctly; mAP50 is the mean Average Precision across all classes with an IoU value of 0.5; mAP50-95 is the mean Average Precision with IoU values ranging from 0.5 to 0.95. In addition, we added a custom metric TopK accuracy, which takes in a user input k, and evaluates the percentage of cases where the actual label is contained in the k most confident predictions. $k = 3$ will be used in this project. This evaluation metric is beneficial for determining the breeds if the dog is a mix. We will also present a confusion matrix to analyze modes of failure. Other metrics, such as training and prediction time, will also be tabulated.

5 Method

This project will focus on enhancing the YOLOv5s model with different attention mechanisms and modified structures.

5.1 Self-Attention Mechanism

The key intuition behind attention mechanisms is to have the model learn to focus on key information that helps the classification task and suppress unimportant information. In the context of dog breed identification, ideally, the model should learn to focus on key features, such as the head and body, while neglecting background information. The attention mechanism used for this project is adapted from Zhang et al.[10]. Given an input feature map x, the query Q, key K, value V, and attention weight can be computed as follows,

$$Q = W_q x, K = W_k x, V = W_v x \Rightarrow A(Q, K, V) = \text{softmax}(QK^T)V \tag{1}$$

where W_q, W_k, and W_v are learnable parameters. A multi-head self-attention is simply a concatenation of multiple self-attention.

5.2 Vision Transformer

A transformer layer consists of a multi-head self-attention module and an MLP module (multi-layer perceptron). Given an input x, the layer can be represented as the following,

$$\begin{aligned} a &= \text{MultiheadAttention}(x) + x \\ y &= W_2(W_1 a) + a \end{aligned} \tag{2}$$

A vision transformer is a block of a sequence of transformer layers. Given some 3D features x, the transformer first flatten the features into $\hat{x}$, then compute the linear projection through

$$y = W\hat{x} + \hat{x} \tag{3}$$

and feeds y into multiple transformer layers consecutively. The output is then reconstructed into a 3D feature.

5.3 Model Structure Modification

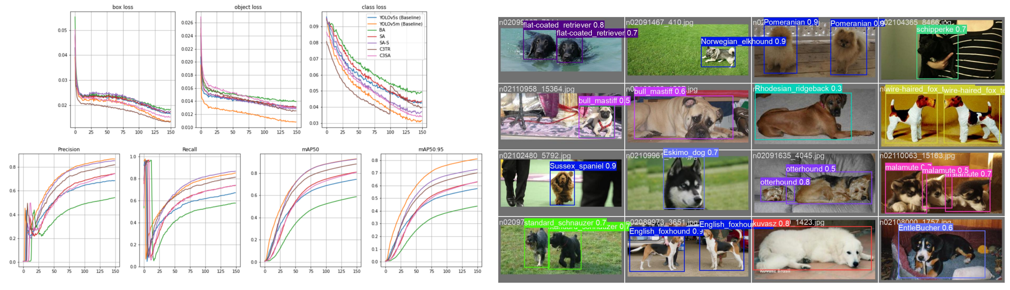
Our first naive attempt was to add self-attention layers immediately after feature extraction, resulting in model BA. The intuition is to have the attention layer focus on crucial features in the image while suppressing unrelated features after feature extraction and pass those crucial features into the next stage for further processing.

We then modified this structure by changing how self-attention layers are inserted into the original structure - insert either to a later stage or an earlier stage. We first tested by inserting the attention layer into a later stage in the neck section, resulting in model NA. The intuition for inserting attention layers into the neck is to keep and combine as many features as possible. Then, at the detection stage, the model will have a wide range of complicated features and can learn to focus on essential features, such as body parts of a dog, and directly output the prediction. Next, we tried to insert the attention layers into an earlier stage by fusing the attention layer into the C3 module in the Backbone, leading to model C3SA. The intuition for fusing attention layers into the C3 module is for future feature extraction layers to extract more detailed features from the features the current layer pays attention to.

Building on those two models, we proposed two final models. The first model, NA-S, has attention layers in the neck section, with more skip connections from the earlier feature extraction stage so that at the detection stage, the model will focus on not only complicated and processed features but also simple features extracted at relatively earlier stages that are lost during the lengthy processing. The second model, C3TR, fuses a vision transformer into the C3 block for multi-head attention. Detailed visualization of each model and individual modules are included in Appendix B.

5.4 Hyperparameter

All models, except the baseline, are trained with their respective maximum possible batch size due to limited GPU memory. To reduce training time, all models use transfer learning with either 10 freezing layers if the backbone is not modified or 4 freezing layers if the backbone is modified. The number of freezing layers is chosen to keep the maximum portion of feature extraction from the pretrained model. Baseline models are trained with a batch size of 16 for the purpose of comparison. All models are trained for 150 epochs using cosine learning rate scheduler and SGD optimizer. Due to the limit of GPU memory, the number of heads in multi-head self-attention is set to 4, and the number of transformer layers in the transformer is set to 1.



(a) (b)

Figure 2: (a) Training loss and metrics on validation set (The discrepancy in Model C3TR's class loss is caused by resuming training and it does not have effect on metrics). (b) Example output on examples (validation) from Figure1a from Model NA-S

6 Results and Analysis

As shown in Figure 2a, among all the models, YOLOv5m has the lowest loss and best performance on the training/validation set. However, as shown in Table 1, while the performances of most models are lowered by approximately 20%, the performance of YOLOv5m is lowered by approximately 30%. This could be explained by model overfitting on the training/validation set. While there are 12,000 images in the training set, each breed of dog only has 100 images, which is not enough for generalization. The large size of YOLOv5m makes it more susceptible to overfitting, especially with a small dataset. A potential improvement is to use a larger dataset or to perform data augmentation on the training set.

As shown in Table 1, the first attempt (Model BA), although slightly larger than the baseline model, yields lower metrics than the baseline model YOLOv5s. It could be explained by the attention layers paying attention to the wrong feature. Specifically, the features may not be meaningful to the model at the early stage of feature extraction. Consequently, when applying attention mechanisms to features, the model may weigh more on features extracted from the background than the dog. When those features are concatenated with other features in the neck and passed onto the head for detection, they may cause harm to the model.

Based on our analysis, we modified our structure and developed Model NA and Model C3SA. Both models perform better than the baseline and yield performance comparable to YOLOv5m. The result proves that adding attention mechanisms to the model can be beneficial. However, they also have some drawbacks. Model NA and Model C3SA can be only trained with maximum batch sizes of 16 and 8, respectively, even though their model sizes are similar or even smaller than YOLOv5s, which is trained with a maximum batch size of 128. This is because the operations in attention weight are more computationally expensive, especially if fused into the C3 module. Therefore, both models' training time per epoch is longer than the baseline.

Building on previous models, we further modified the two model structures and developed Model NA-S and C3TR. Both models perform better than their precedents. Model NA-S is slightly larger than Model NA, but all of its performance scores surpass those of Model NA by approximately 10%, and the training and prediction time is the same as Model NA. Similarly, Model C3TR also outperforms Model C3SA by approximately 10%, but at the cost of smaller batch size, significantly longer training and prediction time. Compared to Model C3TR, Model NA-S performs better overall. It proves that attention mechanisms in the later stage combined with skip connections from the earlier stage of feature extraction are beneficial to the model.

Compared to YOLOv5s, Model NA-S has a significant increase in all performance scores, with an acceptable increase in training and prediction time. Compared to YOLOv5m, Model NA-S also performs better and can predict with less time. Model NA-S also has the highest Top3 accuracy. One example of Top3 being useful is the prediction in the first image of the second row in Figure 2b. Compared to Figure 1a, the model has the correct prediction of breed "pug. However, it is not the most confident one, and the label is partially occluded by the pink label. We also observed that the most common failure is when the dog's color is very close to that in the background, and it can be challenging for the model to detect the presence of a dog. Figure 7 displays some examples of such failures. The confusion matrix further demonstrates such failure modes, as shown in Figure 8. While there are some cases of confusion between breeds, the most common type of error is no detection, as shown in the last row of the confusion matrix. This could be caused by important features not being extracted at the feature extraction stage and potentially be solved by using a larger feature extraction model.

Model	Size	Batch Size	P	R	mAP50	mAP50 -95	Top3	Training Time (per epoch)	Predicting Time (per image)
YOLOv5s	7.2M	16	0.454	0.476	0.461	0.346	0.723	1.5min	2.2ms
YOLOv5m	21.4M	16	0.563	0.547	0.566	0.467	0.78	1.5min	4.5ms
BA	7.6M	16	0.429	0.474	0.442	0.322	0.724	2.5min	3.9ms
NA	7.4M	16	0.569	0.586	0.593	0.448	0.811	2.5min	3.9ms
C3SA	6.7M	8	0.558	0.582	0.595	0.444	0.814	4.5min	5.4ms
NA-S	8.1M	16	0.649	0.632	0.674	0.51	0.859	2.5min	4ms
C3TR	7.3M	4	0.633	0.636	0.664	0.498	0.852	11.5min	17.9ms

Table 1: Comparison of model performance on test set. Best two models are labeled in bold

7 Conclusion and Future Work

We developed, evaluated, and compared several models built upon YOLOv5s using attention mechanisms that outperformed YOLOv5s and YOLOv5m. Our best model, NA-S, shows significant improvement in different evaluation metrics, with an acceptable decrease in speed. For future work, it would be interesting to investigate what each model pays attention to at different stages in visualization. One possible approach is to extract attention parameters from each attention layer or block and reconstruct it to generate a heatmap and visualize what the model is paying attention to in the image. Another potential improvement is to train the models with a larger dataset to reduce overfitting and perform the same modification to larger models such as YOLOv5m.

8 Contributions

Zinan Hu: researches on data processing and implementation of YOLO model using Tensorflow, develop model structures, perform literature review, research on attention mechanism, write report

Yulin Huang: researches on implementation of YOLO model using PyTorch and data processing, develop model structures, perform literature review, research on attention mechanism, produce model structure figures, write report

Genggeng Zhou: develops scripts for data processing, develop model structures, perform model training, develop script for plotting and additional model blocks, modify script for evaluation metrics, manage github repo, write report

References

[1] A. Bochkovskiy, C. Wang, and H. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *CoRR*, vol. abs/2004.10934, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[2] G. Jocher, "YOLOv5 by Ultralytics," 5 2020. [Online]. Available: https://github.com/ultralytics/yolov5

[3] R. B. Girshick, "Fast R-CNN," *CoRR*, vol. abs/1504.08083, 2015. [Online]. Available: http://arxiv.org/abs/1504.08083

[4] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[5] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear convolutional neural networks for fine-grained visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1309–1322, 2018.

[6] 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 (CVPR)*, June 2015.

[7] H. Zheng, J. Fu, T. Mei, and J. Luo, "Learning multi-attention convolutional neural network for fine-grained image recognition," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[8] T.-Y. Zhang, J. Li, J. Chai, Z.-Q. Zhao, and W.-D. Tian, "Improved yolov5 network with attention and context for small object detection," in *Intelligent Computing Methodologies*, D.-S. Huang, K.-H. Jo, J. Jing, P. Premaratne, V. Bevilacqua, and A. Hussain, Eds. Cham: Springer International Publishing, 2022, pp. 341–352.

[9] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[10] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," 2018. [Online]. Available: https://arxiv.org/abs/1805.08318

Appendix

A. Data Processing

The YOLO model requires a text file containing all dog breeds, a text file for normalized bounding box positions (center, height, width) for each image, and a *yaml* file that lists all bog breeds with corresponding numerical labels. The original dataset only provides a *xml* file for each image that defines dog breeds and the coordinates of the upper left and lower right corners of the bounding box in pixels for each image. Therefore, in order to apply the YOLO model in our custom dataset, we developed a function to extract all dog breeds, label them numerically, and save them in a *yaml* file. Another function was also developed to extract information on dog breeds and bounding box positions from each *xml* file. The function reads the dog breed, computes the center, width, and height of the bounding box and normalizes the data, and finally saves all information in a text file for each image.

B. Modified Block and Model Structure

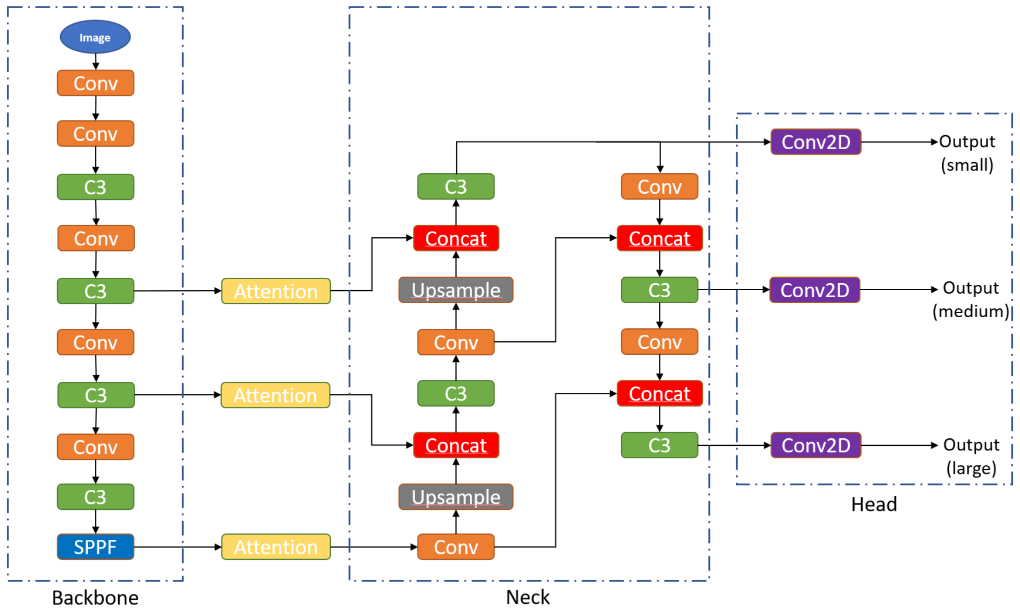


Figure 3: Model BA

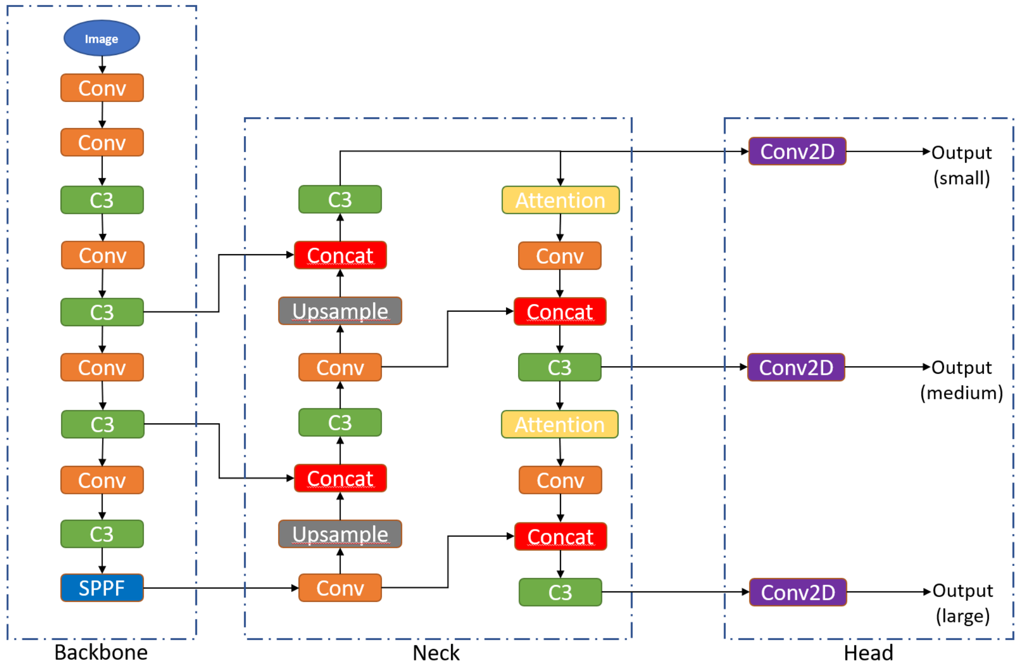


Figure 4: Model NA

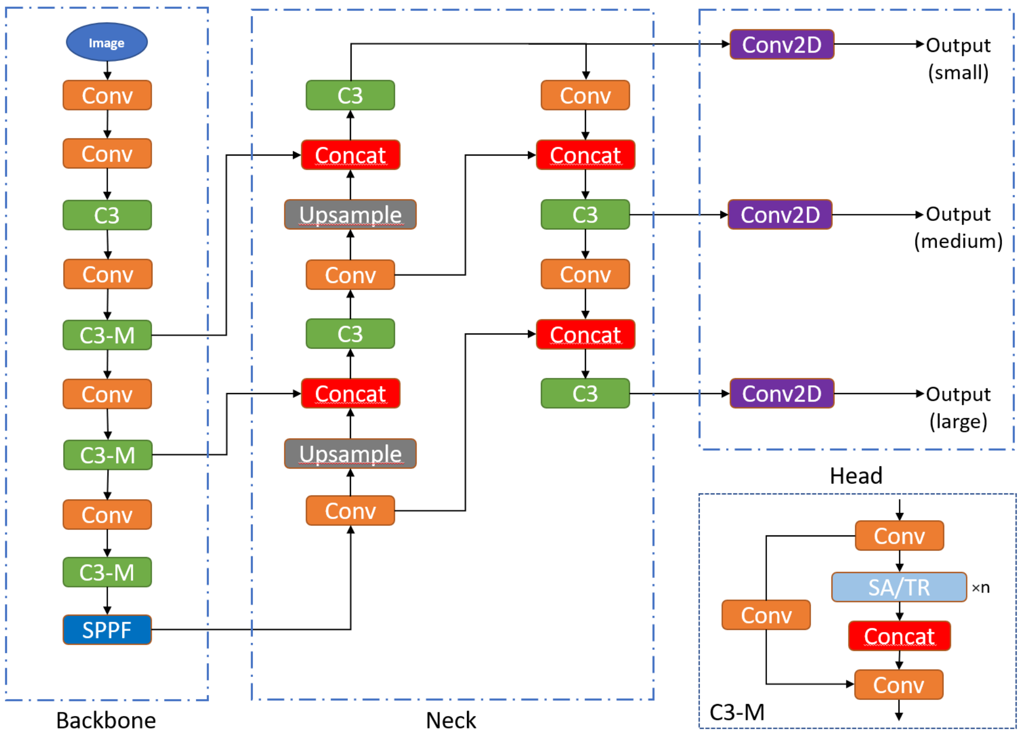


Figure 5: Model C3SA/C3TR

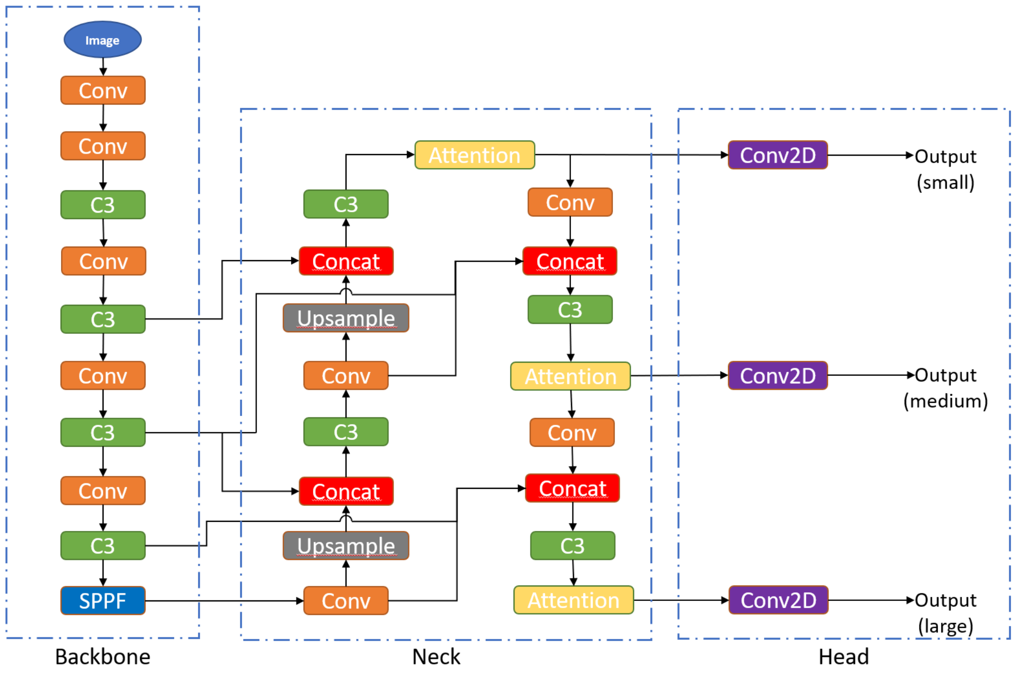


Figure 6: Model NA-S

C. Other Results on Test Set

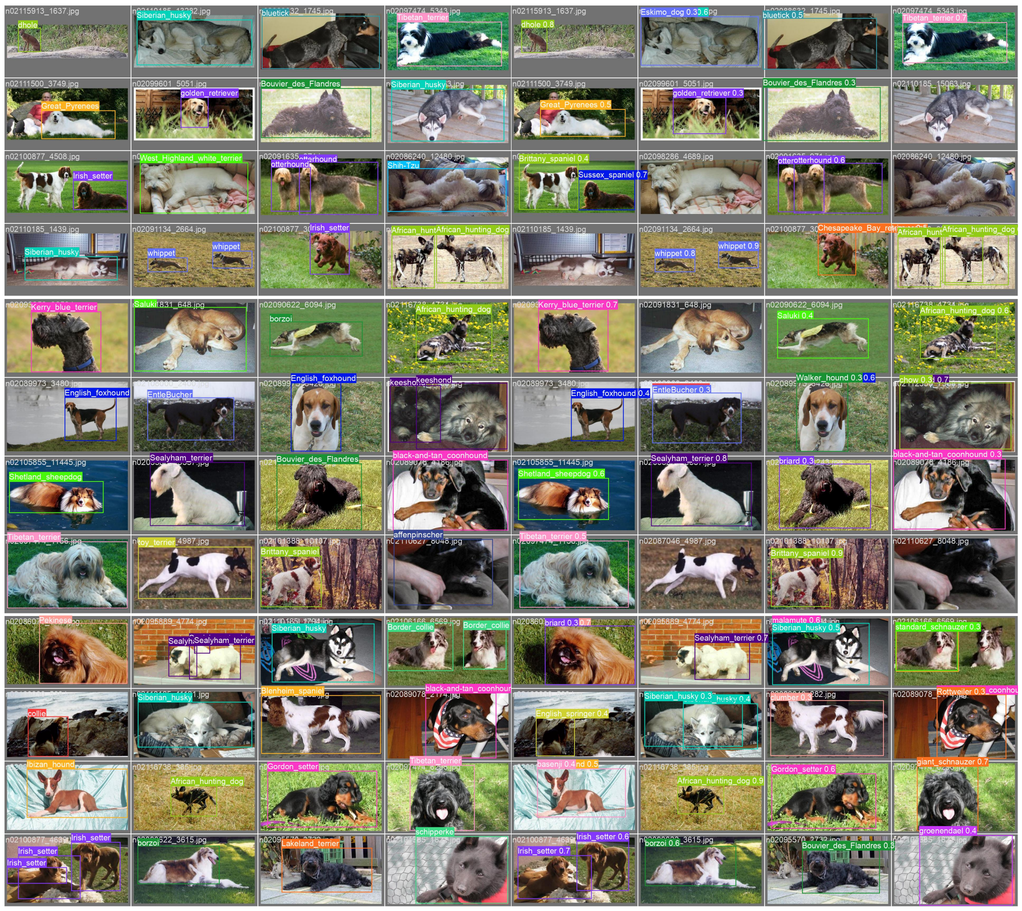


Figure 7: Test set examples. Left: true label. Right: prediction using Model NA-S

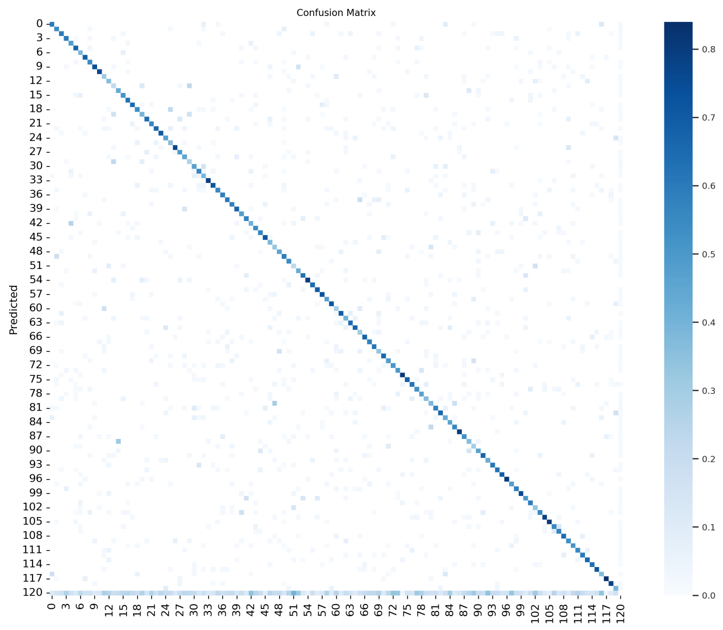


Figure 8: Confusion Matrix for Model NA-S on test set