Where's Whale-do (First Place Solution)

Ammar Ali Mustafa Khalil

1 Basic information for winner announcement

1. Name: Ammar Ali
   Hometown: Syria - Jableh
   Resident: Russia - Saint Petersburg

2. Name: Mustafa Khalil
   Hometown: Portugal - Lisbon

2 Model documentation and write-up

2.1 Who are you (mini-bio) and what do you do professionally?

1. I am a second year PhD student in applied computer science at ITMO University. I have worked on different projects including (Driver monitoring system , Driver assistant system, Text extraction automation) and on ML in specific (Object detection 2D/3D, Image segmentation 2D/3D, image retrieval, classification, OCRs, NERs, Image Captioning, Depth estimation, 3D Construction .. etc).

2. I am a software engineer working at CERN, Switzerland, focusing on Vacuum technologies.

2.2 What motivated you to compete in this challenge?

Well, at first this competition was interesting and fun. Moreover, Ammar started working on image retrieval recently, and since the competition will push to look through the latest SOTA, researches, new techniques and more. we decided to participate to push our knowledge and have fun.

2.3 High level summary of your approach: what did you do and why?

2.3.1 Training experiments and details

At the very beginning, I picked up my baseline from this competition [1], which I participated in 3 months ago for a job interview. I adjusted it for this competition data, started training and sent my first submission. This baseline is quite easy, it uses a simple efficient-b0 with Arcface loss. After making my first submission and making sure that all is good, I started my enhancement plan, modifying the architecture a little, trying different feature extractors, losses and augmentations. I also noticed that on the metadata, we will have information about whether the query image is a top or literal. Due to the unbalance between literal and top images, I decided to train model for top view images only and other models for literal view images. After reading top solutions from [2], I started enhancing by adding GeM pooling, Batch Normalization, Elastic ArcFace. I have tried to train (Convnext, swin, deit, resnet family, Efficientnet), and the best loss obtained was using EfficientNet B5 (B6, B7 wasn't tried because I don't have enough GPU power). I also tried different loss function including multi heads to train model that is able to separate between years (adding such models enhanced the results a little but was removed to keep the submission time limit).In addition, i tried to add MSE loss function between features of the same whale with the ArcFace loss (didn't work), poly loss (best results of it was the same as cross entropy). I was afraid of over-fitting especially that I am training on data with stratified folding (for each whale, I have images in both training and validation), but on the testing data there are completely new whales,and that make a gap between my validation score and the LB (when my validation was above 90, the LB was about 45. Therefore I decided to split the data using group folding to validate over totally new whales. to adjust the hyper-parameters then retrain over the original split (stratified split). That helped me to try tricks and post-processing locally and to be more confidant about my score. My Best single model CV was about 65 @top20 with tricks and about 85 @top1 over completely new whales. PS: All previous details are one only for models trained on top view images. For queries with different years or same year, I have trained some models and used them on the submission but lately I noticed that removing them enhanced my score on testing and LB so they were removed. For literal view Images, I decided to convert them to a top problem based on the fact that if we have a solid model for top1 mAP for literal top cases, then we can use this model to get the best match for a literal query image then use the database first top image as a query image to find the top20. On the other hand, for top-literal queries, I was using the top models to find the first 2 similar query images then use the three query images (the main and the two founded) to get the matching literal images from the database. At the very end, I noticed that this is not eligible, so I removed it from my very last submission. Instead, I used literal models to find the best match literal from the database for the top query image then use this literal database image to

get the top2 matching, and finally used the three images (query image and 2 database images) to find the rest (20). At first, I was training literal models by taking only one literal image and one top image for each whale to keep the data balanced and training was done over all data (no validation). But lately I changed that to another approach so for each top image in the training data, I took a random literal image then I trained the model so that for each batch and each label there are two images one top and another literal and loss is computed as arcface sum of both the literal features and top features, and that gave me a nice boost on the LB.

2.3.2 Tricks

1. Test Time Augmentation (TTA), horizontal flip worked for me any thing else didn't work for LB but worked for CV, So I just kept the horizontal flip.

2. Use database images for re-ranking. For each query image, I found the best match top2 then computed the similarities for each image (query image, database images) with other database image.

3. The similarities then added using weighted average with the following weights (0.4, 0.4, 0.2) Which are chosen according to the validation results of one fold.

4. The similarities are raised to the power 7 before adding (using weighted average) This will definitely help if it is not necessary that the database has 20 or more images. and it gives a nice boost to my local testing but I don't know if it did enhance the LB or made it worse even though I kept it [3].

2.3.3 Summary

Models trained:

1. EfficientNet-b5 with noisy student initial weights (5 Folds) over top view image only (stratifiedkfold over labels).

2. EfficientNet-rw-M (3 Folds 0 - 2 - 4) over top view image only (stratifiedkfold over labels).

3. EfficientNet-b5 with noisy student initial weights (4 Folds 0 - 1 - 2 - 4) over top and literal view images (stratifiedkfold over labels). Only whales that have a corresponding literal and top images on the database.

Loss Functions:

1. Elastic ArcFace (All models)

2. Cross Entropy (All models)

Training augmentations:

1. Resize (512x512)
2. HorizontalFlip (p = 0.5)
3. One of (Sharpen, ToGray, CLAHE p = 0.3) for illumination stability p = 0.5
4. ShiftScaleRotate (p = 0.5)
5. ColorJitter (p = 0.5) for illumination stability
6. Triangle (truncate white triangle on corners) used only while training EfficientNet-rw-M models

2.4 Solution Diagram

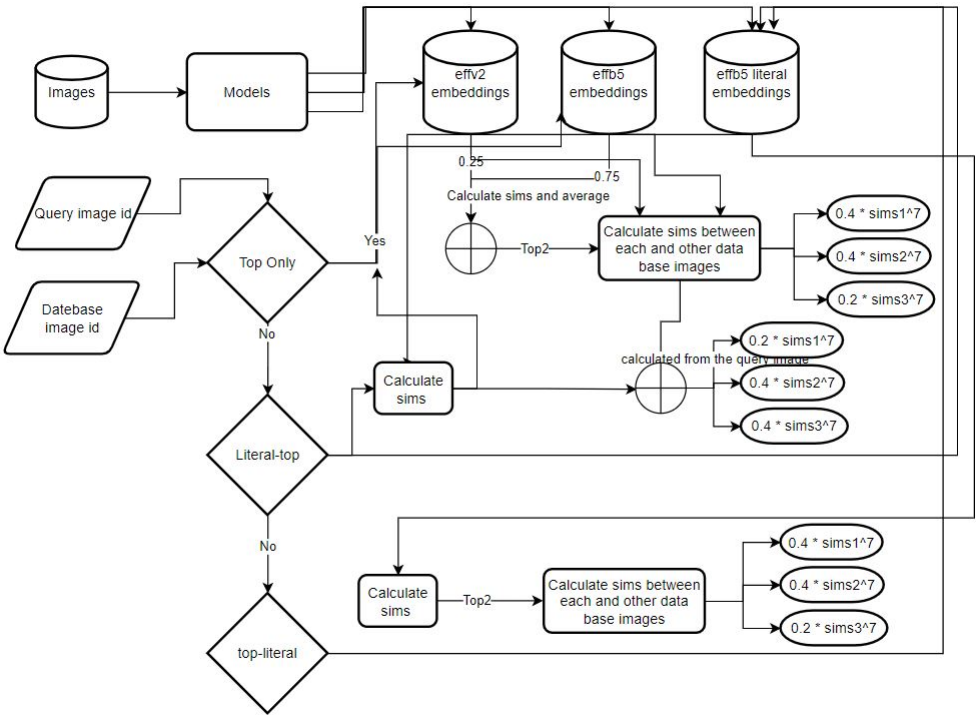

Figure 1: Inference (Solution Architecture)

3 Do you have any useful charts, graphs, or visualizations from the process?

The same on the DATA EXPLORATION AND BENCHMARK provided by DrivenData nothing else.

4 Copy and paste the 3 most impactful parts of your code and explain what each does and how it helped your model.

4.1 Inference

```
if literal_top:
    # use literal models to get the top1 then usetop models to find top2 from the data base
    # Calculate the similarities for between top2 on database with other database images
    qry2_embedding = emb2.loc[[top1.index[0]]]
    qry2_embedding_eff2 = emb2_eff2.loc[[top1.index[0]]]
    sims2 = cosine_similarity(qry2_embedding, _db_embeddings2)[0]
    sims3 = cosine_similarity(qry2_embedding_eff2, _db_embeddings2_eff2)[0]
    sims2 = (0.25 * sims3 + 0.75 * sims2)
    sims2[sims2 < 0 ] = 0
    sims2[sims2 > 1 ] = 1
    top2 = pd.Series(sims2, index=_db_embeddings2.index).sort_values(0, ascending=False).head(2)
    qry2_embedding = emb2.loc[[top2.index[1]]]
    qry2_embedding_eff2 = emb2_eff2.loc[[top2.index[1]]]
    sims3 = cosine_similarity(qry2_embedding, _db_embeddings2)[0]
    sims4 = cosine_similarity(qry2_embedding_eff2, _db_embeddings2_eff2)[0]
    sims3 = (0.25 * sims4 + 0.75 * sims3)
    sims3[sims3 < 0 ] = 0
    sims3[sims3 > 1 ] = 1

elif top_literal:
    # Use Literal models to get top2 literals from other literals on the database and
    # Calculate the similarites with other database images to these top2
    qry2_embedding = embeddings.loc[[top1.index[0]]]
    sims2 = cosine_similarity(qry2_embedding, _db_embeddings)[0]
    qry2_embedding = embeddings.loc[[top1.index[1]]]
    sims3 = cosine_similarity(qry2_embedding, _db_embeddings)[0]
    sims3[sims3 < 0 ] = 0
    sims3[sims3 > 1 ] = 1

else:
    # Use top models to get top2 from database and calculate similarites between these
    # top2 and other dataset images
    qry2_embedding = embeddings.loc[[top1.index[0]]]
    qry2_embedding_eff2 = embeddings_eff2.loc[[top1.index[0]]]
    sims2 = cosine_similarity(qry2_embedding, _db_embeddings)[0]
    sims3 = cosine_similarity(qry2_embedding_eff2, _db_embeddings_eff2)[0]
    sims2 = 0.75 * sims2 + 0.25 * sims3
    qry2_embedding = embeddings.loc[[top1.index[1]]]
    qry2_embedding_eff2 = embeddings_eff2.loc[[top1.index[1]]]
    sims3 = cosine_similarity(qry2_embedding, _db_embeddings)[0]
    sims4 = cosine_similarity(qry2_embedding_eff2, _db_embeddings_eff2)[0]
    sims3 = 0.75 * sims3 + 0.25 * sims4
    sims3[sims3 < 0 ] = 0
    sims3[sims3 > 1 ] = 1
```

Figure 2: Inference Re-ranking

Using database images to get the best 20 matches has improved the results significantly. The main reason is that my models has a high top1 score. Therefore less error rate when getting only the best two matches. Then using multiple representation for the same whale to get the best 20 matches will for sure enhance the results. To make it more simple imagine that each image is represented as a point in an N dimensional space . So using only query image to get the best matches will simply find the closest 20 points. But if we took the top 2 then we will have a triangle on that space instead of the point and we will be looking for the closes points to the center of that triangle.

4.2 Training

```
def train_fn(dataloader,model,criterion,optimizer,device,scheduler,epoch):
    model.train()
    loss_score = AverageMeter()
    criterion2 = torch.nn.MSELoss()
    tk0 = tqdm(enumerate(dataloader), total=len(dataloader))
    for bi,d in tk0:
        batch_size = d[0].shape[0]
        images = d[0]
        targets = d[2]
        images2 = d[1]
        images = images.to(device)
        images2 = images2.to(device)
        targets = targets.to(device)
        optimizer.zero_grad()
        output, feats1 = model(images,targets)
        output2, feats2 = model(images2,targets)
        loss = criterion(output,targets) + criterion(output2,targets)
        loss.backward()
        optimizer.step()
        loss_score.update(loss.detach().item(), batch_size)
        tk0.set_postfix(Train_Loss=loss_score.avg,Epoch=epoch,LR=optimizer.param_groups[0]['lr'])
    if scheduler is not None:
            scheduler.step()
    return loss_score
```

Figure 3: Training Literal models

Passing two different images at once to the model at each epoch to calculate the loss, this technique helped the model to balance the performance between literal and top view images.

```
x = self.backbone(x)
x1 = self.pooling(x).view(batch_size, -1)
x2 = self.pooling(x).view(batch_size, -1)
x3 = self.pooling(x).view(batch_size, -1)
x = torch.cat((x1, x2, x3), dim=1)
x = self.bn(x)
x = self.ln(x)
if self.use_fc:
    x1 = self.dropout(x)
    x1 = self.bn(x1)
    x1 = self.fc(x1)
return x
```

Figure 4: Training models replicating the pooling output

As you can see, I was taking the output of the pooling layer three times then concat it and let the linear layers narrow down to the number of features, This kind of same multi pooling helped the linear layer to adjust the weights and minimize the loss even more.

5 Please provide the machine specs and time you used to run your model.

Specifications:

1. CPU: Intel Core i9 10th Generation
2. GPU: RTX 3090, VRAM 24 GB
3. Memory (RAM): 128 GB
4. OS: Linux

Training Duration:

1. EfficientNet-rw-M: 220 minutes for a single fold (3 folds) in total about 11 hours.
2. EfficientNet-b5 top: 180 minutes for a single fold (5 folds) in total about 15 hours
3. EfficientNet-b5 literal: 180 minutes for a single fold (4 folds) in total about 12 hours.

Inference Duration:
It took about 2 hours and 45 minutes on driven data env and specifications. The models took about 2 hours and 45 minutes for post-processing. If TTA is removed the models will take about one hour and 15 minutes.

6 Anything we should watch out for or be aware of in using your model (e.g. code quirks, memory requirements, numerical stability issues, etc.)?

Nothing in particular. It is stable and reproducible on different machine on my side. It is better to train fold by fold and not to make a for loop for training.

7 Did you use any tools for data preparation or exploratory data analysis that aren't listed in your code submission?

No, but I used another training scripts with different splitting approach to validate and adjust the hyper-parameters then I used the provided training code with these hyper parameters/augmentations.

8 How did you evaluate performance of the model other than the provided metric, if at all?

Using the provided metric (mAP) and the elastic Arcface loss .

9 What are some other things you tried that didn't necessarily make it into the final workflow (quick overview)?

1. Polyloss
2. Convnext, Deit, swin , resnet200D, resnest, resnext (feature extractors)
3. multi head models to distinguish between years
4. Specific models for different years images
5. Higher image sizes
6. heavy augmentations

10 If you were to continue working on this problem for the next year, what methods or techniques might you try in order to build on your work so far? Are there other fields or features you felt would have been very helpful to have?

1. In case i have better GPU power, i will try tranformers (swin, deit) again because they will probably get better results or larger models of efficient-Net Series
2. In case the time limit is open i will add the models which trained to distinguish between years (two head models) it will give a small boost as well
3. For sure the current inference is not perfect some techniques could be changes, some threshold and that may give a nice boost to the score.
4. Train CNN feature extractor with multiple crops of a single image and use RNNs or self attention mechanism to add the features on different levels.
5. Re annotate the data with bboxes and train object detection model to detect certain regions then retrain feature extractors for specific regions instead of full image input.
6. annotate masks for scars, and use it as metadata to help the models to converge or use a two stages training approach.

References

[1] "https://www.kaggle.com/competitions/where-is-the-toilet-the-right-one"

[2] https://www.kaggle.com/competitions/happy-whale-and-dolphin/discussion/320310

[3] https://www.youtube.com/watch?v=VxNDH6qLZ_Q&ab_channel=NVIDIADeveloper