Predicting Popularity of Shelter Pet Photos with Vision Transformers and CNNs

Aalok Patwa, Rishabh Mandayam, Trang Dang

1. Abstract

In this project, we build and train image models to predict the popularity of images of shelter pets. This task is important for shelters, since whether or not a pet gets adopted is largely influenced by the way the pet is presented photographically. The task is also important for machine learning broadly, as it tests image models on being able to recognize human sentiment; that is, this task goes beyond just identifying objects in images, instead demanding that models predict how humans might react to an image, which is more nuanced. Our main contributions are the construction of vision transformers and CNNs for this task, and an analysis of their strengths and weaknesses. We build models with competitive RMSE compared to the Kaggle leaderboard, compare the effectiveness of image models for feature extraction and fine-tuning, and determine that regression approaches lead to better performance than classification.

2. Introduction

Our task is to predict the popularity of shelter pet images from the website of a large Malaysian animal shelter, called Petfinder.my. Our dataset was taken from a recent Kaggle challenge, called the Petfinder.My Pawpularity Contest. This dataset contains two different forms of inputs: 1) a dataset of images of pets in the shelter, and 2) a table providing summary metadata for each of the images in the image dataset. To be more specific, the table (henceforth referred to as the metadata table) contains binary features describing the nature of each image, such as whether the pet's eyes are facing forward. The dataset contains 9,912 images and the metadata table has 9,912 rows. The output is the Pawpularity Score, a numerical value between 1 and 100 provided as a column in the metadata table. We are given a ground-truth Pawpularity score for each of the 9,912 images in the dataset. According to the Kaggle challenge website, the Pawpularity score was derived from statistics of Internet traffic to each pet's webpage. To eliminate noise, the score was normalized across different pages and platforms, and traffic from bots was filtered out.

We will be training on 70% of these 9,912 images, reserving 10% for a validation set to use during each epoch of neural network training, and 20% for a hold-out test set. To evaluate the final performance of our models, we will calculate root mean squared error (RMSE) between our models' predictions and the ground-truth Pawpularity scores.

3. Background

1st Place - Winning Solution - Full Writeup.
https://www.kaggle.com/competitions/petfinder-pawpularity-score/discussion/301686. The winning Kaggle submission for this dataset uses multiple pretrained models, including CNNs and vision transformers, to extract embeddings from the image set. It then used these embeddings to train a support vector regressor (SVR) to predict Pawpularity score. Once the

final dataset is built, the support vector regressions achieved a competition winning RMSE of 16.95.

The issue with this prior work was its black box nature. It does not analyze or explain the choice of framing this problem as a regression task. It does not mention why the models chosen as feature extractors had such good representation learning ability. Our project is meant to provide reasoning by directly comparing different problem framings and models, as well as experimenting with modifications to the dataset.

4. Summary of Our Contributions

Contribution in Analysis:
We investigate three major questions:

1. How should we best approach this as a deep learning problem? Namely, we investigate framing this as classification (where our target is a one-hot vector and we output class probabilities) or regression (where our target is a scalar Pawpularity score).
2. How should we utilize neural networks' ability to perform representation learning? We implement both vision transformers and CNNs, and experiment with a two-task fine-tuning pipeline, which involves first fine-tuning on a dog and cat breed dataset and then on our dataset.
3. How can we modify our dataset to improve performance? We experiment with correcting the skew and imbalance in the ground-truth Pawpularity score distribution.

5. Detailed Description of our Contributions

5.1 Methods

Fine-Tuning Vision Transformer and CNN using Regression and Classification Approach
We fine-tune a Vision Transformer using both a regression and classification approach [Ze Liu, 2021]. We used timm (PyTorch Image Models) to create a generic vision transformer model, vit-base-patch16-224 (meaning 16x16 patches on 224x224 input dimensions) with pretrained weights from ImageNet. For the regression approach, we connected the head of the transformer, which contained 768 nodes, with a single output node with sigmoid activation. We then normalized our ground truth Pawpularity scores to the range [0, 1]. We used Binary Cross Entropy loss, with a learning rate of 1e-3. For the classification approach, we connected the head of the transformer to 100 output nodes with a softmax activation at the end. We trained using Categorical Cross Entropy loss with a learning rate of 1e-3. We had experimented with other learning rates, but they were not as successful.

We also train a CNN with a ResNet50 backbone for both a regression and classification approach [Kaiming He, 2016]. First, we build a regression ResNet by adding a fully connected layer with one output neuron. We scale the Pawpularity score to between 0 and 1 to be consistent with batch-normalized activations in other layers. We then preprocess the images such that they match the inputs from ImageNet. Then, we implement our model by downloading ResNet50 weights and modifying the average pool and fully connected (fc) part as is shown in Appendix Exhibit 1(a). In addition, we build a ResNet for classification. For this part, we choose to divide the Pawpularity score into 100 fine-grained classes. We change the last fully connected layer to output 100 values, and add a softmax layer at the end, as is shown in Appendix Exhibit 1(b). We used cross entropy loss to train. We use an Adam optimizer with a learning rate of 1e-3 for both models.

Transfer Learning with Image Classification and Breeds Classification (Two-tasks ResNet)

We download a model that has been trained on ImageNet and further fine-tune our model with a cats and dogs breeds dataset. We hypothesize that such a breed classification task will prepare the model with more understanding about cats and dogs images, leading to better performance.

For this breed classification task, we use the Oxford Cats and Dogs Breeds Classification Dataset. This data set includes 7,349 images of 37 different breeds of cats and dogs with different sizes, pose, and lighting. We download ResNet50 and its ImageNet weights, and, with this as backbone, we replace its average pooling layer with a convolutional layer, a max-pooling layer, and a dense layer, and then replace the fully connected step with a combination of dense layers, drop out layers, and a final soft-max layer (Refer to Appendix Exhibit 2 for the architecture). We used the Adam optimizer and cross-entropy loss. After training this network on breed classification over 6 epochs and using a 70:30 train-test split, we achieve an accuracy rate of 97.5%.

Then, we fine-tune this network on Pawpularity images and scores. First, we take this network and replace the last fully connected layer so that it can output one value (Refer to Appendix Exhibit 3 for the architecture). Next, we preprocess Pawpularity scores to be between 0-1 instead of 0-100 and we preprocess the images so that they match ImageNet statistics. Finally, we train the model for 5 epochs with an 80:20 train-test split.

To evaluate the breed classification task impact on the model's performance, we also fine-tune a ImageNet-pretrained ResNet50 with the same architecture (save the last fully-connected layer) on our task alone. We use the same optimizer, number of epochs, and train-test split.

CNN and ViT Feature Extraction without Fine-Tuning

We then followed the advice of the Kaggle winning submission, which suggested extracting features from the last layer of pre-trained image models and using them to train a support vector regressor. For this experiment, we did not finetune the image models; rather, we just took the embedding layer of the image models directly. We then used the activations of the embedding layer as features for the support vector regressor, which was trained with labels normalized to the [0, 1] range. We chose the optimal regularization level for the support vector regressor using 5-fold cross validation. We then evaluated the performance of the regressor on the held-out test set. We also extracted features from an additional vision transformer architecture, called the Swin transformer, to see if it would perform better than the traditional transformer.

Correcting Skewed Pawpularity Score Distribution

A consistent theme across our experiments has been the role of the scores' skewed distribution in influencing our models. We observed that because the Pawpularity scores in the dataset are right-skewed (Appendix Figure 6), models naturally gravitate towards predicting more modal values between 20 and 50. We experiment with fixing this class imbalance. We use the imbalanced-learn package to implement Random Oversampling of less frequent scores in the dataset, which turns the distribution of Pawpularity scores into a uniform distribution.

5.2. Experiments/Results

Fine-Tuning Vision Transformer and CNN using Regression and Classification Approach

We observe that the classification approach results in significantly worse performance than the regression approach for both Vision Transformer and CNN. Furthermore, the Vision Transformer outperforms the CNN overall.

	Vision Transformer	**CNN**
Classification RMSE	20.9	24.3
Regression RMSE	18.2	19.8

Table 1: Vision Transformer and ResNet's test performance for the classification approach and regression approach

According to Table 1, when treated as a classification task, both the Vision Transformer's RMSE, 20.9, and the CNN's RMSE, 24.3, is higher than simply predicting the average score (RMSE 20.59), so they have no predictive value here. However, when the task is treated as a regression task, CNN's has a RMSE of 19.8, so there's some predictive value in this case. The regression Vision Transformer was able to attain a reasonable RMSE of 18.2, beating the CNN.

We can attribute the success of the regression approach over the classification approach to the fact that regression allows for distance-based loss functions. In the classification domain, cross-entropy loss treats the classes (represented as length-100 vectors) as unordered bins, meaning that an output that predicts score 1 when the ground truth is 99 is just as bad as an output that predicts score 85. By contrast, regression approaches allow for losses like MSE and Binary Cross Entropy, which penalize predictions based on the actual distance of the prediction from the ground truth.

Pre-training Regressor ResNet with Breeds Classification Data set

The ResNet that has been pretrained on both ImageNet and Oxford Cats and Dogs Breeds Classification performs significantly better than the vanilla Resnet that has only been trained on ImageNet. When the network is trained on recognizing cats and dogs breeds in addition to classifying objects from ImageNet, the RMSE test loss decreases about 1 to 1.5 points in 5 epochs (Refer to Appendix Table 4). According to Table 2 below, this additional task results in a jump in the performance on the test data set with a RMSE of 17.69 from 20.377 on the held-out test dataset.

Vanilla ResNet	**Two-tasks ResNet**
20.377	17.69

Table 2: Test RMSE for ResNet pretrained on ImageNet vs ImageNet & Cats and Dogs Breeds

When we study some small examples (Appendix Figure 4 and 5) and plot the relationships between scores predictions for the two Resnets (Appendix Figure 1 and 2), we learn that Vanilla Resnet output a much wider range of scores, resulting in predictions that are very different from the original scores (Appendix Figure 1), while Two-tasks ResNet predicts scores that are relatively close to the 25 to 50 range (Appendix Figure 2). In Appendix Figure 3, we can see that the score difference between original scores and predicted scores for Vanilla ResNet's predictions has a higher mean, 75-percentile, and higher outliers, reflecting how Vanilla Resnet's variations result in worse predictions than Two-tasks ResNet. The performance difference confirmed our hypothesis that additional information about cats and dogs is indeed beneficial to the score prediction tasks.

<u>CNN and ViT Raw Feature Extraction</u>

Results on the held-out test set for various model architectures, including CNNs and ViT's, are shown in the table below.

Base Model	**Test RMSE**
ResNet50	17.98
16x16 Patch ViT	17.75
SWIN Transformer	17.87

Table 3: Test Result for Various Resnet, Patch ViT, and SWIN Transformer

These results are competitive, and show the effectiveness of image models as pure feature extractors. We hypothesize that training on a large corpus like ImageNet gives ResNet50, the 16x16 patch Vision Transformer, and the Swin Transformer, the ability to find regions of interest on images, such as the faces of pets. The 16x16 patch Vision Transformer slightly outperformed the other model architectures, which could be a testament to the superiority of vision transformer architectures.

<u>Correcting Skew/Imbalance in Pawpularity Scores</u>

We obtain worse RMSE on the held-out test set after correcting skew. This can be explained by the jump from turning the originally skewed distribution of scores to a uniform distribution. The test set follows the same distribution as the training set, so it has the same skew towards scores in the range 20-40. After correcting skew, the predictions made by an SVR with vision transformer embeddings as features demonstrate a more normal distribution of predictions (Appendix Figure 7), which is problematic as the original data is not normal.

6. Compute / Other Resources Used

We wrote all of our code and ran our models in Google Colab, making use of the GPU when possible. We did not use any other resources for compute and rarely trained models on our own hardware.

7. Conclusions

Outcomes: We discovered that framing this problem as a regression task was more successful than framing it as a classification task. This result was unexpected, as in our previous checkpoints we hypothesized that classification worked better. In hindsight, regression allows for distance-based loss functions, which encourage model predictions to be close to the ground truth better than categorical cross-entropy on a one-hot vector. We also observed that a fine-tuned vision transformer outperformed a fine-tuned ResNet50, giving evidence to recent talk about the superiority of vision transformers. We saw that both vision transformers and CNNs are successful at performing feature extraction, since support vector regressors trained on embeddings produced by their base architectures were quite accurate. We were successful with our experiment to fine-tune ResNet50 on a cat and dog breed dataset before our task, which shows how a subtle domain adaptation before final fine-tuning can improve the representation learning capacity of image models. Finally, we learned that correcting class imbalance is not always necessary if we seek to be objective and make accurate predictions, since the real world data likely follows the same distribution as the training set.

For the future: We would like to conduct more specific inferential experiments to compare the predictions made by vision transformers and CNNs. Specifically, we would like to uncover the attention maps of the vision transformer, which would detail what patches in the image the transformer had the highest attention for. This would be very revealing in terms of how attention mechanisms learn. We would compare it to the feature maps of CNNs.

Ethical Considerations: The models we develop allow shelters to make better predictions about the success of pet images on their websites. If deployed, it would lead to hyper-optimal choices of what photos to post online, which could possibly skew reality. For example, organizations' online presence could revolve around the images that AI predicted to be the most successful at garnering interest. This could be used in perverse ways to trick the broader public.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770-778. 2016.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. "Swin transformer: Hierarchical vision transformer using shifted windows." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10012-10022. 2021.

Broader Dissemination Information

We would like our report to be published on the website along with our names.

We have no other links to publish.

Appendix

```
model.avgpool = nn.Sequential(
     nn.Conv2d(2048,512,3),
     nn.MaxPool2d(2),
     nn.Flatten())
 model.fc = nn.Sequential(
     nn.Linear(2048, 512),
     nn.ReLU(),
     nn.Dropout(0.25),
     nn.Flatten(),
     nn.ReLU(),
     nn.Dropout(0.25),
     nn.Linear(512, 10),
     nn.Softmax(dim=1))
```

```
model.avgpool = nn.Sequential(
     nn.Conv2d(2048,512,3),
     nn.MaxPool2d(2),
     nn.Flatten())

 model.fc = nn.Sequential(
     nn.Linear(2048, 512),
     nn.ReLU(),
     nn.Dropout(0.25),
     nn.Flatten(),
     nn.ReLU(),
     nn.Dropout(0.25),
     nn.Linear(512, 1))
```

(a) Classification ResNet *(b) Regression ResNet Model*

Exhibit 1: Pytorch Implementation of Classification and Regression ResNet

```
def get_breeds_classifier_model():
 model = models.resnet50(weights=models.ResNet50_Weights.IMAGENET1K_V1)
 for param in model.parameters():
   param.requires_grad = False

 model.avgpool = nn.Sequential(
     nn.Conv2d(2048,512,3),
     nn.MaxPool2d(2),
     nn.Flatten())

 model.fc = nn.Sequential(
     nn.Linear(2048, 512),
     nn.ReLU(),
     nn.Dropout(0.25),
     nn.Flatten(),
     nn.ReLU(),
     nn.Dropout(0.25),
     nn.Linear(512, 37),
    nn.Softmax(dim=1))

loss_fn = nn.CrossEntropyLoss()
return model.to(device), loss_fn
```

Exhibit 2: ResNet Architecture for Cats and Dogs Breeds classification task

```
model.fc = nn.Sequential(
    nn.Linear(2048, 512),
    nn.ReLU(),
    nn.Dropout(0.25),
    nn.Flatten(),
    nn.ReLU(),
    nn.BatchNorm1d(512),
    nn.Linear(512, 1),
    nn.Sigmoid())

loss_fn = nn.BCELoss()
```

(a) Two-tasks Resnet

```
model.fc = nn.Sequential(
    nn.Linear(2048, 512),
    nn.ReLU(),
    nn.Dropout(0.25),
    nn.Flatten(),
    nn.ReLU(),
    nn.Dropout(0.25),
    nn.Linear(512, 1))


loss_fn = nn.MSELoss()
```

(b) Vanilla ImageNet Resnet

Exhibit 3: ResNet Architecture for Pawpularity Image Scores Task

ResNet Pretrained on ImageNet Image Classification	**ResNet Pretrained on ImageNet Image Classification and Breeds Classification**
20.053102	19.036683
21.793355	19.820112
20.814711	19.281529
19.932881	18.862201
20.103507	18.978621

Table 4: Test Loss for ResNets pretrained on ImageNet or ImageNet and Cats and Dogs Breeds During Training

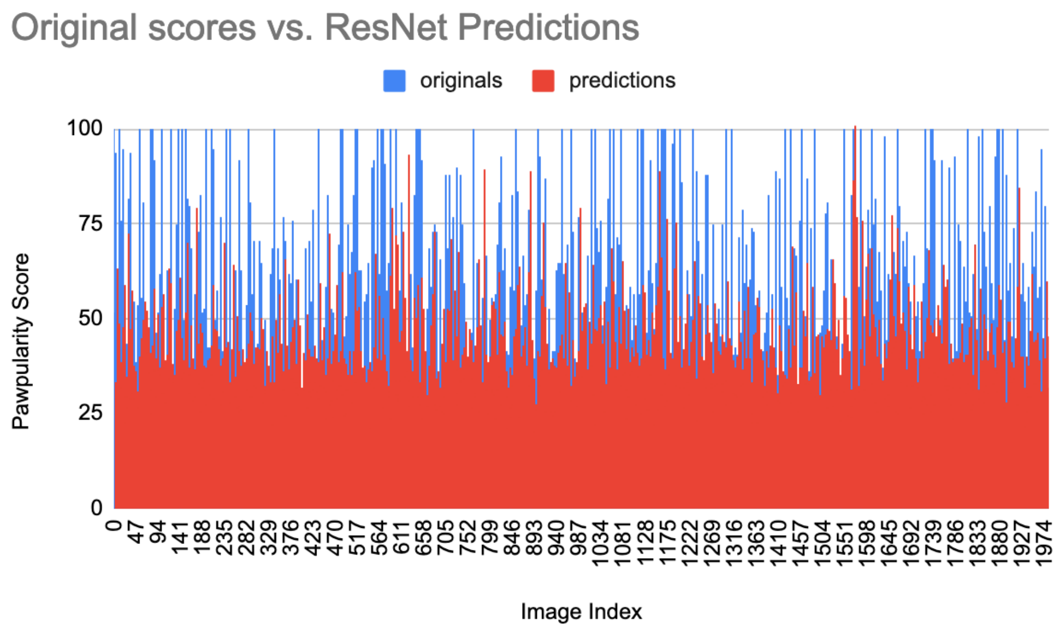


Figure 1: A line graph that shows the original score (blue) and Vanilla Resnet's predictions for each image in the test data set.

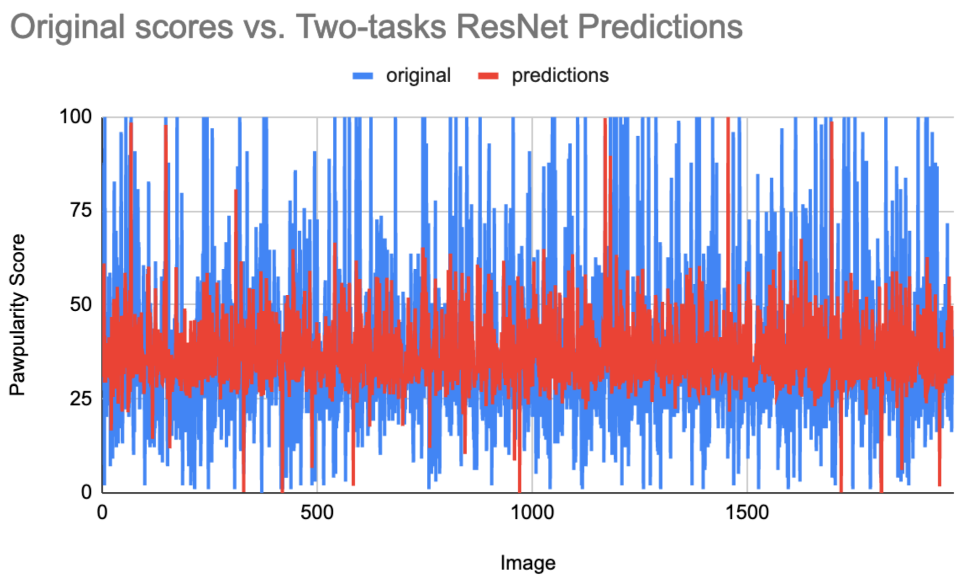


Figure 2: A line graph that shows the original score (blue) and Two-tasks Resnet's predictions for each image in the test data set.

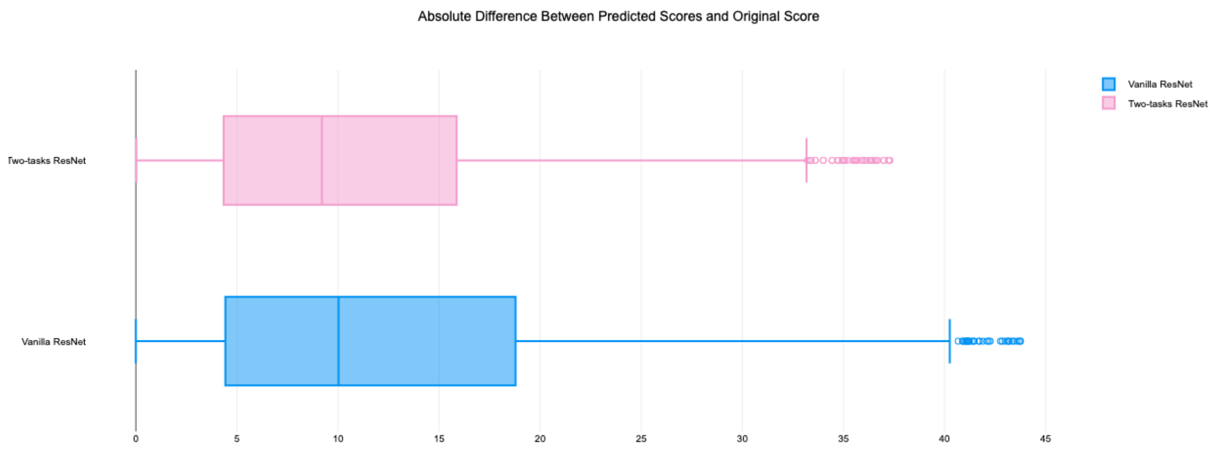
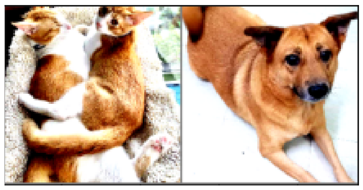
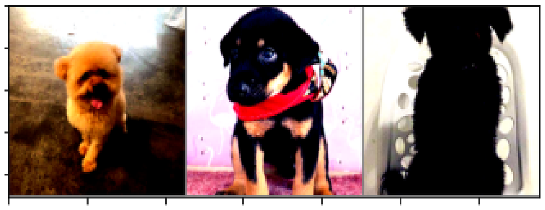


Figure 3: Box plots of the absolute difference between the original scores and the scores predicted by Vanilla Resnet (blue) and Two-tasks Resnet (pink).



Predicted scores: 25.72, 31.68
Real scores: 28.28, 32.32

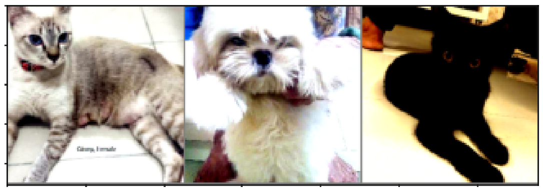
Good Predictions



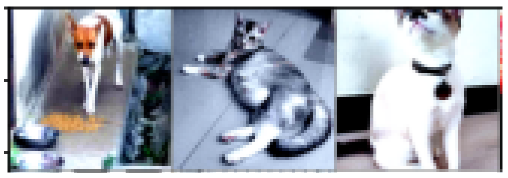
Predicted scores: 100.89, 51.66, 49.78
Real scores: 100.0, 91.91, 100.0

Bad Predictions

Figure 4: Good and Bad Score Predictions From Vanilla ResNet



Predicted scores: 42.42, 98.50, 37.39
Real scores: 41.41, 100.0, 41.41



Predicted Scores: 36.86, 45.43, 46.72
Real Scores: 28.28, 66.67, 38.38

Figure 5: Good and Bad Score Predictions From Two-tasks ResNet

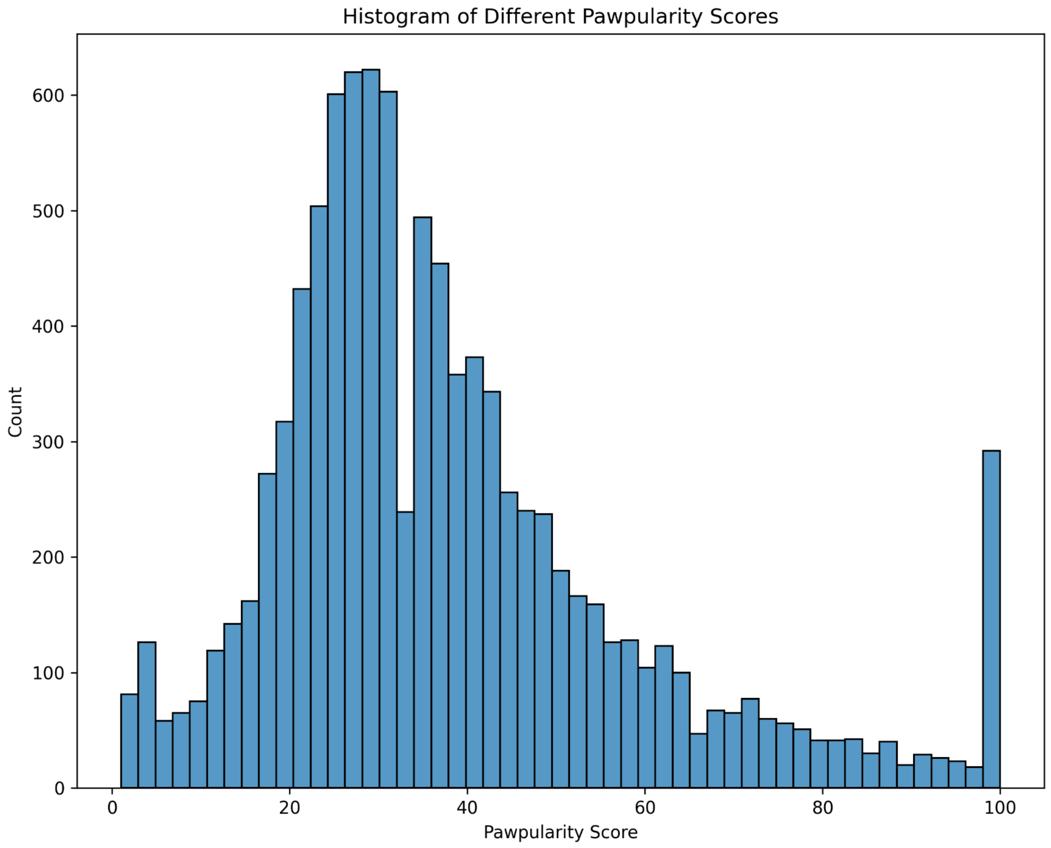


Figure 6: Histogram showing distribution of ground-truth Pawpularity scores in the dataset.

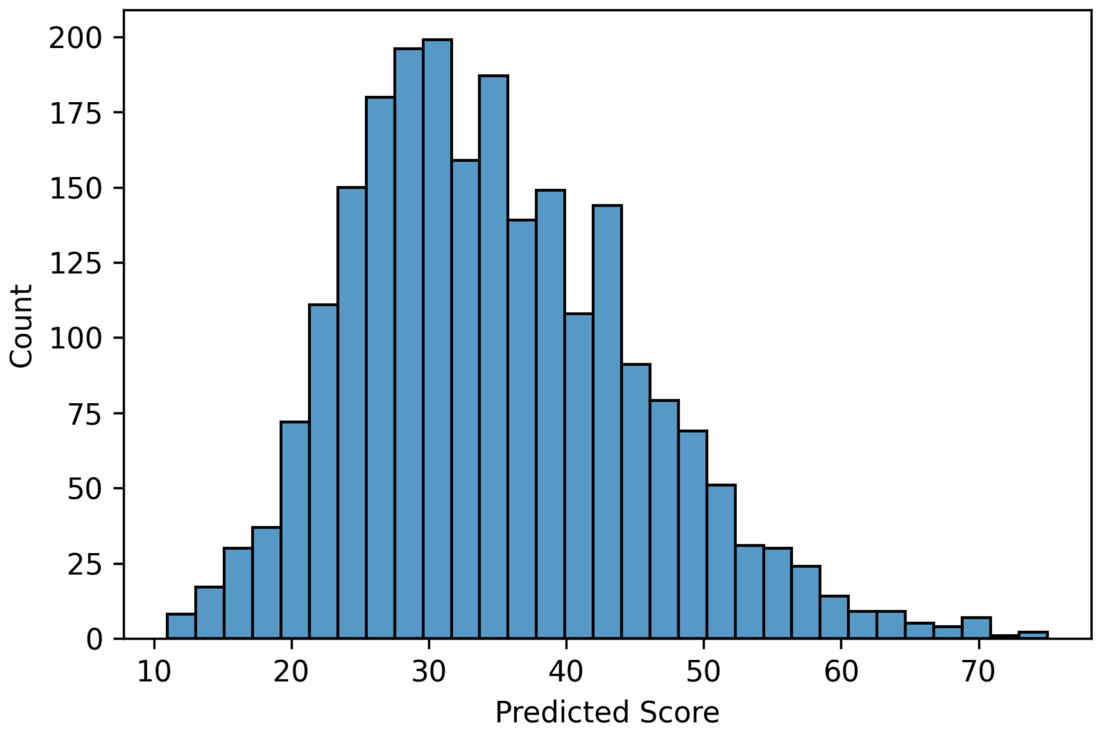


Figure 7: Histogram showing distribution of predicted Pawpularity scores after correcting class imbalance. The distribution is flatter and more Gaussian than the distribution of ground truths.