HOW "PAWPULAR" IS YOUR PET?

USE DEEP LEARNING TO SCORE PET CUTENESS

YIFAN GU

SEUNG-MI JEON

Overview

01
Background

"Pawpularity" Prediction

02
CNN Modeling

- Hyperparameter Tuning
- Adaptive CNN
- Transfer Learning

03
CNN + XGBoost

- Pretrained CNN
- Train with XGBoost
- Hyperparameter Tuning

04
Best Model

Key Takeaways from Best Model

Background: PetFinder.my Pawpularity Prediction

- Predict the “Pawpularity” (cuteness) score of pet photos to see how likely they attract adopters
- Regression Problem – predict a continuous “Pawpularity” score ranging from 0 to 100

Data Overview

Data Provided

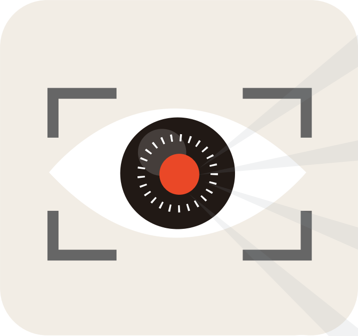
- Image Data: Over 9,900 pet photos
- Metadata: Features such as focus, eyes, face, and more

Evaluation Metric

Root Mean Squared Error (RMSE)

CNN
MODELING

Initial Data Preprocessing Summary



Load, Resize, Convert Images

- Resize all images to 256×256 pixels, ensuring uniform input size
- Convert images to NumPy arrays (float32) for processing

Standardization (Z-score Normalization)

- Normalize images using Z-score normalization to standardize input distribution

Dataset Splitting

- Randomly shuffle data for better generalization
- Split into 70% training, 15% validation, and 15% test

Convert to TensorFlow Dataset

- Apply batching (size 32) and shuffling

Hyperparameter Tuning Guide for CNN Optimization

Hyperparameter	Effect	Tuning Strategy
Number of Conv Layers	Feature extraction depth	Increase layers for complex tasks but avoid overfitting
Number of Filters	Controls feature extraction capacity	Start small (16-64), increase (128-512) for complex features
Filter Size	Feature scale detection	3×3 or 5×5 or 7×7, larger filters for high-level features
Padding	Output size control	“Same” to preserve dimensions, “Valid” to shrink
Pooling Type	Feature retention	Max pooling for best feature retention
Learning Rate	Step size in weight update	Use learning rate decay (0.001 to 0.0001)
Optimizer	Convergence speed	Adam (default), SGD with momentum for fine control
Batch Size	Gradient estimation	Start with 32-64, increase for speed
Number of Epochs	Training iterations	Use Early Stopping
Dense Units	Capacity to learn representations	Start with 64-256 , decrease in deeper layers to prevent overfitting
Dropout	Overfitting control	Use 0.3-0.5 in fully connected layers
L2 Regularization	Prevents large weights	Default: 0.0001 to 0.001
Data Augmentation	Generalization	Use rotations, flips, color jitter

Hyperparameter Tuning Process: Model 1-7

Model Name	Conv Layers (# Filters per Layer)	Kernel Sizes	Dense Layer Size	Learning Rate	Dropout Rate	MSE	MAE	Runtime(min)
Model 1	64 → 32 → 16	3×3	64	0.001	0.4	460.45	16.31	31
Model 2	16 → 32 → 64 → 128	3×3	64	0.001	0.4	477.35	16.71	15
Model 3	16 → 32 → 64 → 128	3×3, 3×3, 5×5, 7×7	64	0.001	0.4	446.38	15.63	30
Model 4	16 → 32 → 64 → 128	3×3, 3×3, 5×5, 5×5	128	0.001	0.4	463.86	16.08	30
Model 5	16 → 32 → 64 → 128	3×3, 3×3, 5×5, 5×5	64	0.01	0.4	504.14	16.49	50
Model 6	16 → 32 → 64 → 128	3×3, 3×3, 5×5, 7×7	64	0.001	0.5	433.42	15.62	36
Model 7	16 → 32 → 64 → 128	3×3, 3×3, 5×5, 7×7	64	0.001	0.6	457.78	15.43	40

→ **Model 6 provides the best balance of accuracy and efficiency**

- Lowest MSE (433.42) and one of the lowest MAE (15.62)
- Balanced runtime (36 minutes)
- Improved generalization with dropout = 0.5
- Maintained a stable learning rate (0.001)

Why Data Augmentation Failed – Potential Problem of Image Resizing

Model 8:
Data Augmentation

- MSE: 520.10
- MAE: 16.12

Possible Issue:
Poor Image Quality

- Data augmentation magnify image distortions
- May loose key features

Problem:
Image Resizing Creates Distortions

- **Unified Input (256×256)**
- Image ratio distortions made some pets appear stretched

Fixing Image Resizing Issues – Better Preprocessing = Better Performance

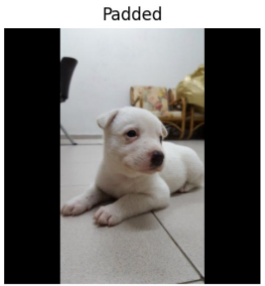
01

02

Smart Resizing with Padding

- padding (black) is added around the image
- Maintains original aspect ratio, prevents stretching

- Improves MSE (424.92) and MAE (15.21) over Model 6



Adaptive CNN with Global Average Pooling

- Handles variable input sizes dynamically
- Usually require more training data for stability (data augmentation applied)

- Achieves best MSE (414.42) and MAE (15.17)

Transfer Learning with EfficientNetB0

Transfer Learning & Benefits

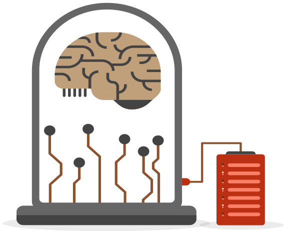
- use a pretrained model that has already learned general image features from large datasets (e.g., ImageNet)
- Speeds up training with pre-learned feature extraction
- Reduces training time and improves accuracy with limited data (~7000 training data)

EfficientNetB0

- Balances accuracy & efficiency better than larger models like ResNet or VGG
- Lightweight but powerful—ideal for fine-grained cuteness recognition

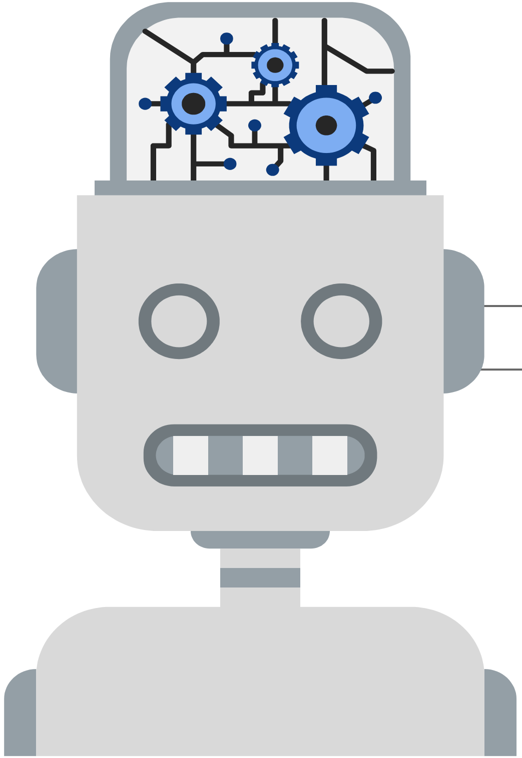
Model Performance

Model	MSE	MAE	Observations
Model 6	433.42	15.62	Best model during hyperparameter tuning
Adaptive CNN	414.42	15.17	Improved upon Model 6
EfficientNet	411.34	15.26	Best accuracy with strong generalization



HYBRID MODELING

Problem Breakdown



Practical Constraints

01

Computing Resource Limit
Need for reducing time
and risk of overfitting

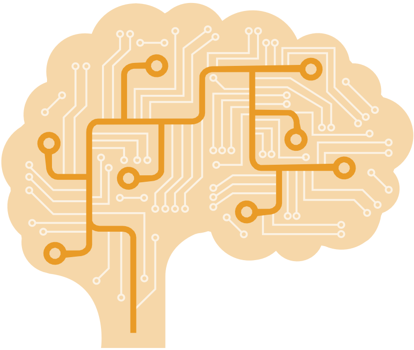
Low Interpretability

02

Build a clearer path to
interpret the relationships
between numerous features

Breakthrough: Hybrid Model

Let's combine two different models for getting synergy!



01 Image Processing

02 Feature Extraction & Merging Data

03 Training and Validation with Merged Data

Why Use Pretrained CNN?

Instead of training a CNN from scratch, use a pretrained ResNet-50 model

- A pretrained model has already learned general visual features (edges, shapes, textures) from a large dataset (ex: ImageNet)

- Pretrained CNNs capture general features that often transfer well to new, domain-specific tasks
- Improves model performance when the new dataset is relatively lacks diverse examples.

- ResNet-50 has been trained on ImageNet
- It allowed the model to extract high-level visual features from pet images

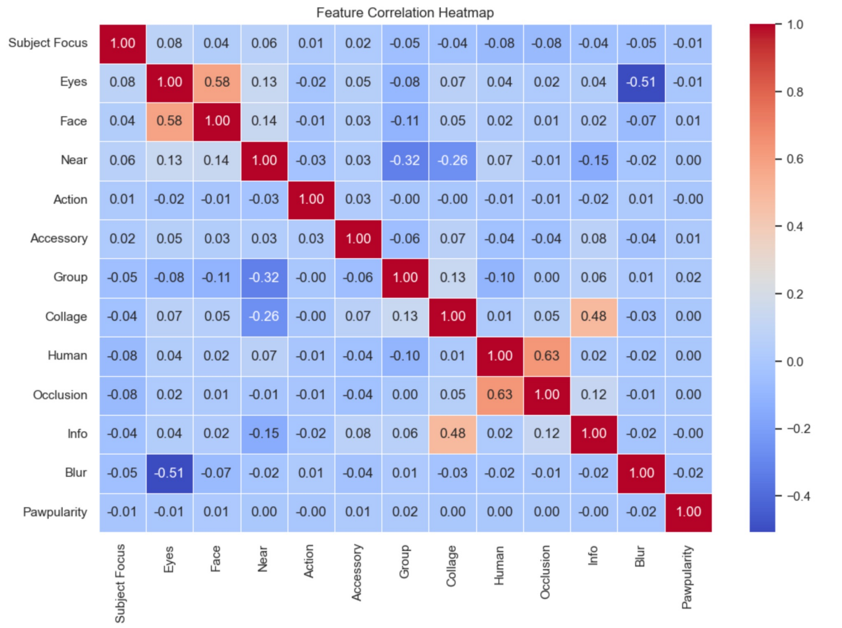
Re-explore Metadata

Linearity Check

| Pawpularity does not show linear relationships with any other features

| All element of metadata is not significantly correlated to Pawpularity

| Feature Correlation values with Pawpularity are close to zero(0)



CNN + XGBoost Hybrid Model

01 Pretrained CNN

ResNet50

02 Feature Extraction

Merge Extracted Features to Metadata

03 Define XGBoost

Non-linear Patterns

04 Fine Tuning

Randomized Search

05 Feature Engineering

PCA, Top100 Features

06 Comparison Results

Complete!

Extract Features from Train Images

Pretrained ResNet-50 as Feature Extractor

- Make a 2048-dimensional feature vector for each image

- For image processing, resize and normalize the image data
- Convert the image to tensor format for model input

- Extracts feature vectors using the pretrained ResNet-50 model
- Converts extracted CNN feature vectors into a structured DataFrame

ResNet-50 Advantages

- Compared to other pretrained models(ex: EfficientNet or Vision Transformer (ViT)), ResNet's structure is intuitive and widely supported by deep learning frameworks, making it easier to implement and modify

- ResNet effectively captures low-, mid-, and high-level visual features (edges, corners, textures), allowing it to generalize well when applied to new datasets (such as pet images)

- ResNet strikes a good balance between implementation complexity, hardware requirements, and overall performance.

- ResNet introduces residual connections, which help alleviate the vanishing gradient problem in deep networks, improving training stability and performance, especially for deeper architectures

Integrate into Tree-based Model

Minimizing Mean Squared Error(MSE) and Mean Absolute Error(MAE)

- Perform 10 random searches across the defined hyperparameter space
- Uses 5-fold cross validation(cv=5) to ensure robust performance evaluation
- Parallel Processing(n_jobs=2) speeds up computation

Best MSE: 352.0318 > RMSE: 18.76

Best MAE: 13.92

```
param_dist = {
    'n_estimators': [100, 300],
    'max_depth': [3, 5],
    'learning_rate': [0.05, 0.1],
    'subsample': [0.8, 1.0],
    'colsample_bytree': [0.8, 1.0]
}
```

```
'subsample': 0.8,
'n_estimators': 100,
'max_depth': 3,
'learning_rate': 0.05,
'colsample_bytree': 0.8
```

Key Takeaways

01 Solve Practical Constraints

Minimize the need for time and the risk of overfitting

02 Leverage the Strengths of Both Approaches

Combining CNN and XGBoost to take both strengths

03 Easier Interpretability

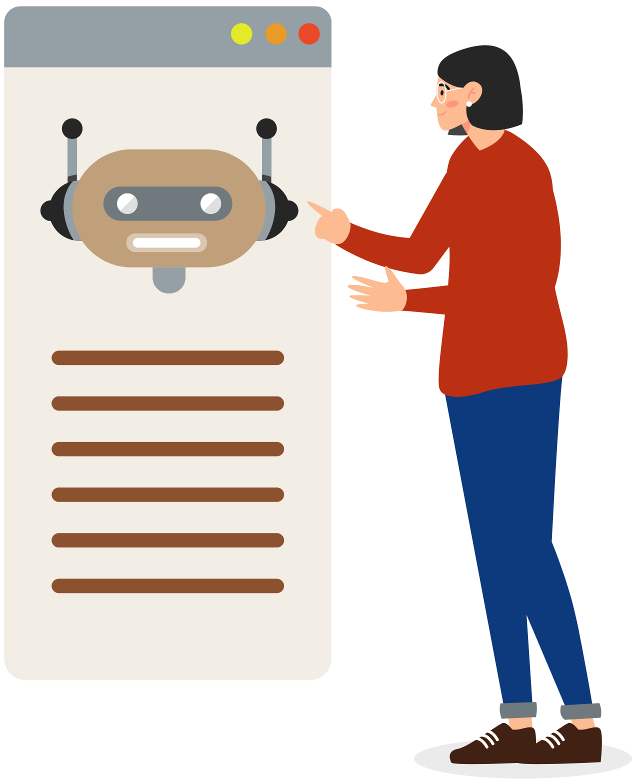
Yield a clearer path to interpret which features drive the model's predictions and see the feature importance

04 Reduced Training Complexity

Easily tune without re-architecting a full learning pipeline

05 Fast Experimentation and Tuning

Quickly test different XGBoost settings for experimentation



“

CNN + XGBoost hybrid provides a sweet spot

Deep CNN for automated visual feature extraction
XGBoost for robust regression on both image & metadata

Lower computational cost, and faster experimentation

”

THANK YOU