Application of Transfer Learning & Automated Machine Learning in Image Classifiers for Dog Breeds

Group 17
Brandon Attai, Tahsin Chowdhury, Kelten Falez, Timothy Mok, Aron Saengchan, and Yong Jun Zhu

ENEL 645: Data Mining & Machine Learning
Final Project Presentation

April 6, 2022

Problem Statement & Objective

- **Problem Statement:**
  - Develop a multiclass image classifier for a set of dog breed images
- Apply various machine learning and deep learning methods
  - Convolution neural networks (CNNs) using transfer learning
  - Ensemble methods
  - Automated machine learning (AutoML)
  - Gradient-weight class activation mapping (Grad-CAM)

Dataset

10,222
images

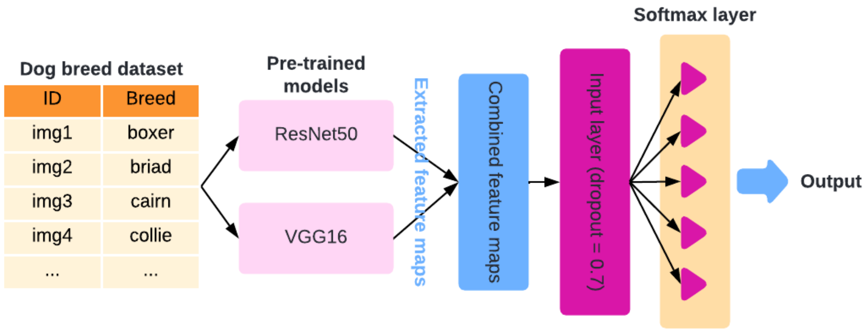
120
classes

Transfer Learning Models

- Two CNN models were used as the primary architectures for this dataset:
  1. ResNet50
  2. VGG16
- Transfer learning methods were applied by replacing the fully connected section of the pre-trained model with a custom output layer of dense and dropout layers
- Models were tested using:
  - Softmax activation function
  - Categorical cross entropy loss function
  - Learning rate of 0.0001
  - Early stopping and patience

Ensemble Model

- Ensemble learning methods were employed to further improve the validation accuracy
- Adapted a method to extract and combine features from each optimized ResNet50 and VGG16 model
- Objective was to acquire a more diverse range of features in order to better classify the dog breed images



AutoML Models

- Two different AutoML options were explored:
    1. AutoKeras (Keras) – All model architectures
    2. AutoGluon (Amazon) – All model architectures
- Compare the benefits and drawback between the two options
    - Fit time
    - Validation accuracy

Deep Learning Model Results

- ResNet50 model
  - Val. accuracy: 62.1% → 79.8%
  - Still overfitting
- VGG16 model
  - Val. accuracy: 53.1% → 68.7%
  - Slightly underfitting
  - Low accuracy
- Ensemble model
  - Low training accuracy and high validation accuracy (89.3%)
  - Short training time

	Training accuracy (%)	Validation accuracy (%)	Training time (s)
Baseline ResNet50	**99.9**	**62.1**	**800**
Best ResNet50	**95.4**	**79.8**	**893**
Baseline VGG16	**99.9**	**53.1**	**808**
Best VGG16	**65.7**	**68.7**	**2034**
Ensemble	**42.6**	**89.3**	**14**

AutoML Results (AutoKeras)

- Model architecture

```
Layer (type)                Output Shape              Param #
=================================================================
input_1 (InputLayer)       [(None, 224, 224, 3)]     0

cast_to_float32 (CastToFloa (None, 224, 224, 3)      0
t32)

normalization (Normalizatio (None, 224, 224, 3)      7
n)

random_translation (RandomT (None, 224, 224, 3)      0
ranslation)

random_flip (RandomFlip)   (None, 224, 224, 3)       0

efficientnetb7 (Functional) (None, None, None, 2560)  64097687

global_average_pooling2d (G (None, 2560)             0
lobalAveragePooling2D)

dense (Dense)              (None, 120)               307320

classification_head_1 (Soft (None, 120)              0
max)

=================================================================
Total params: 64,405,014
Trainable params: 64,094,280
Non-trainable params: 310,734
```

Model	Training accuracy (%)	Validation accuracy (%)	Training time (s)
AutoKeras	**97.5**	**82.9**	**2604**
AutoGluon	**85.5**	**89.9**	**2083**

AutoML Results (AutoGluon)

- Model hyperparameters

```
Finished, total runtime is 2080.38 s
{ 'best_config': { 'batch_size': 16,
                   'dist_ip_addrs': None,
                   'early_stop_baseline': -inf,
                   'early_stop_max_value': inf,
                   'early_stop_patience': 10,
                   'epochs': 100,
                   'final_fit': False,
                   'gpus': [0],
                   'lr': 0.01,
                   'model': 'resnet50',
                   'ngpus_per_trial': 8,
                   'nthreads_per_trial': 128,
                   'num_workers': 4,
                   'searcher': 'random',
                   'seed': 468,
                   'time_limits': 7200},
  'total_time': 2005.747457742691,
  'train_acc': 0.8546385017421603,
  'valid_acc': 0.8993157380254154}
```

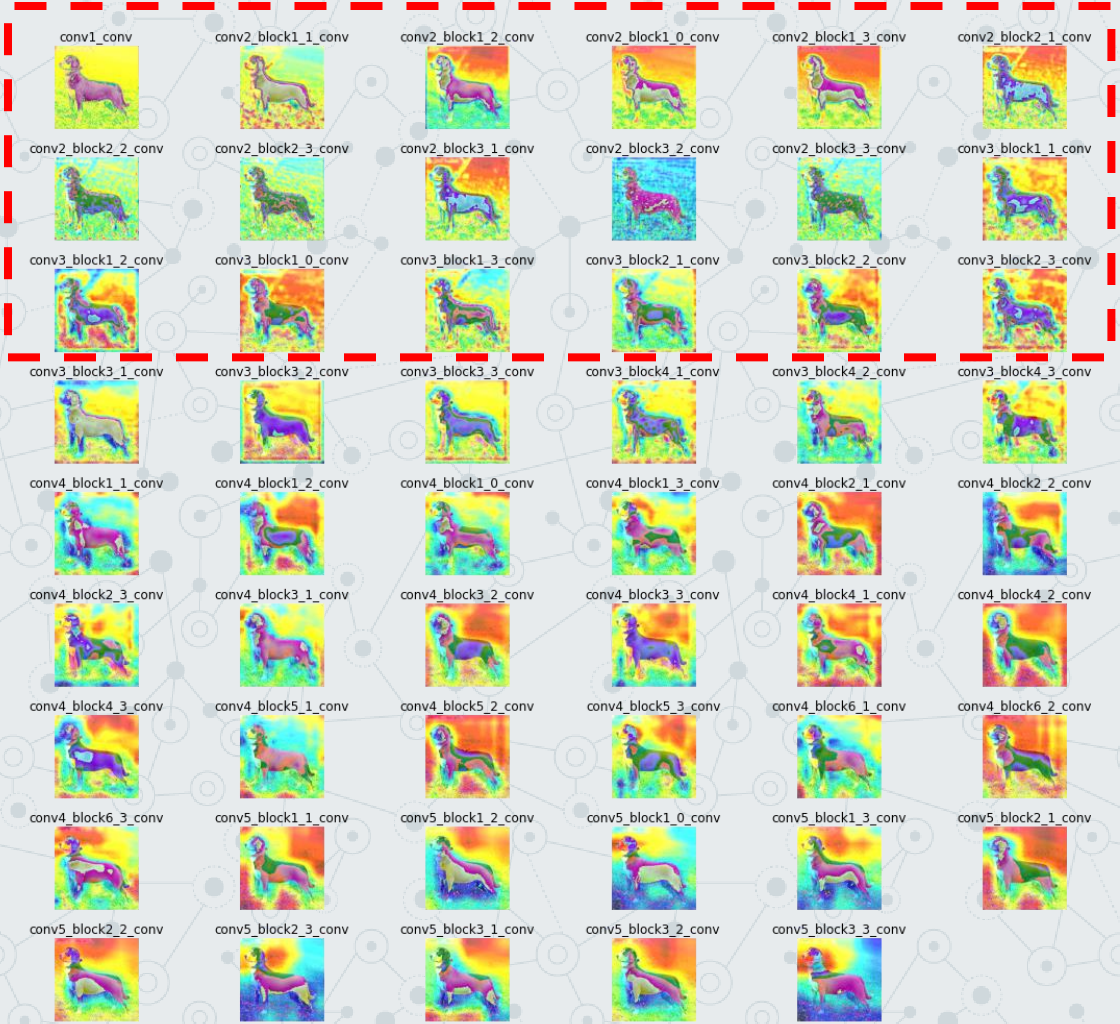
Model	Training accuracy (%)	Validation accuracy (%)	Training time (s)
AutoKeras	**97.5**	**82.9**	**2604**
AutoGluon	**85.5**	**89.9**	**2083**

Grad-CAM Analysis

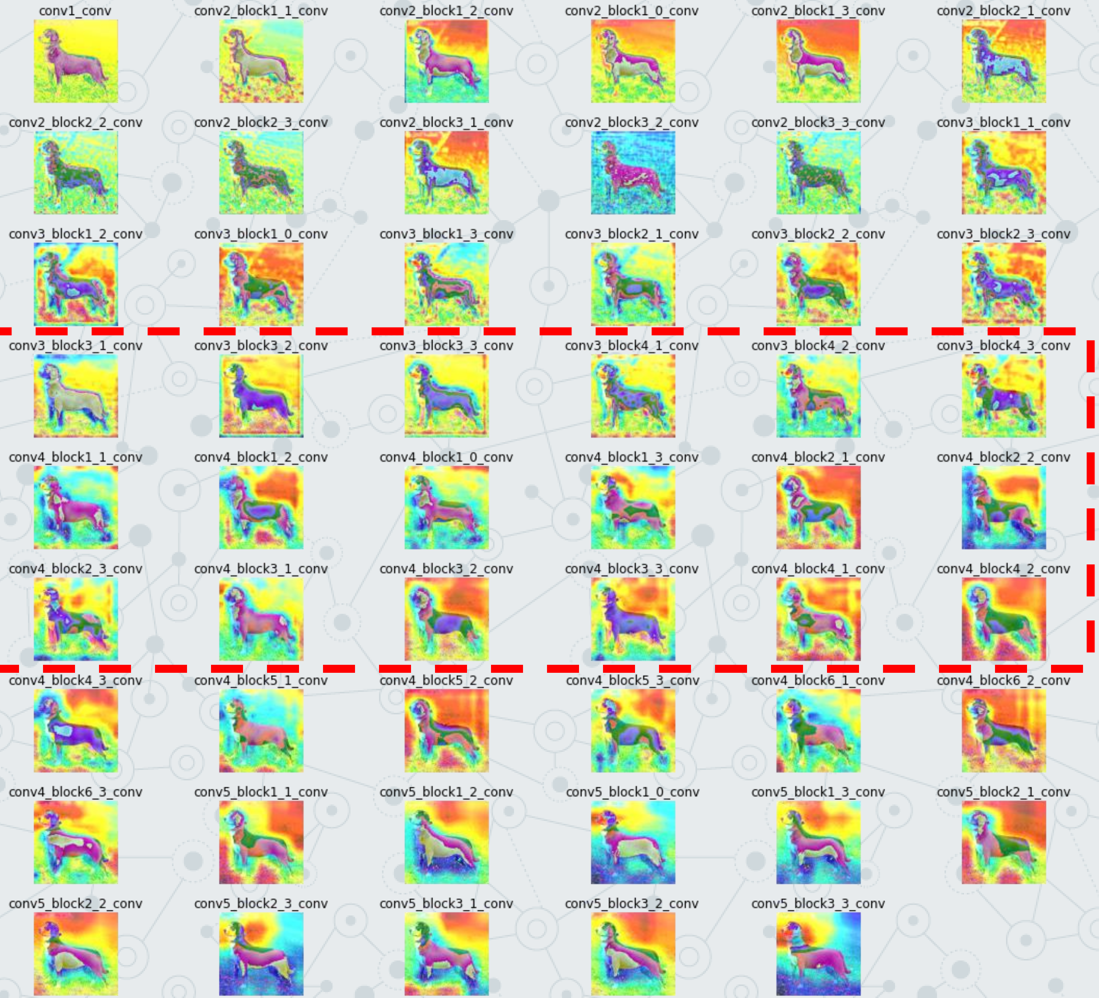
- Grad-CAM analysis was used to make the models more interpretable
- Highlights the regions of an image that was deemed important by the model

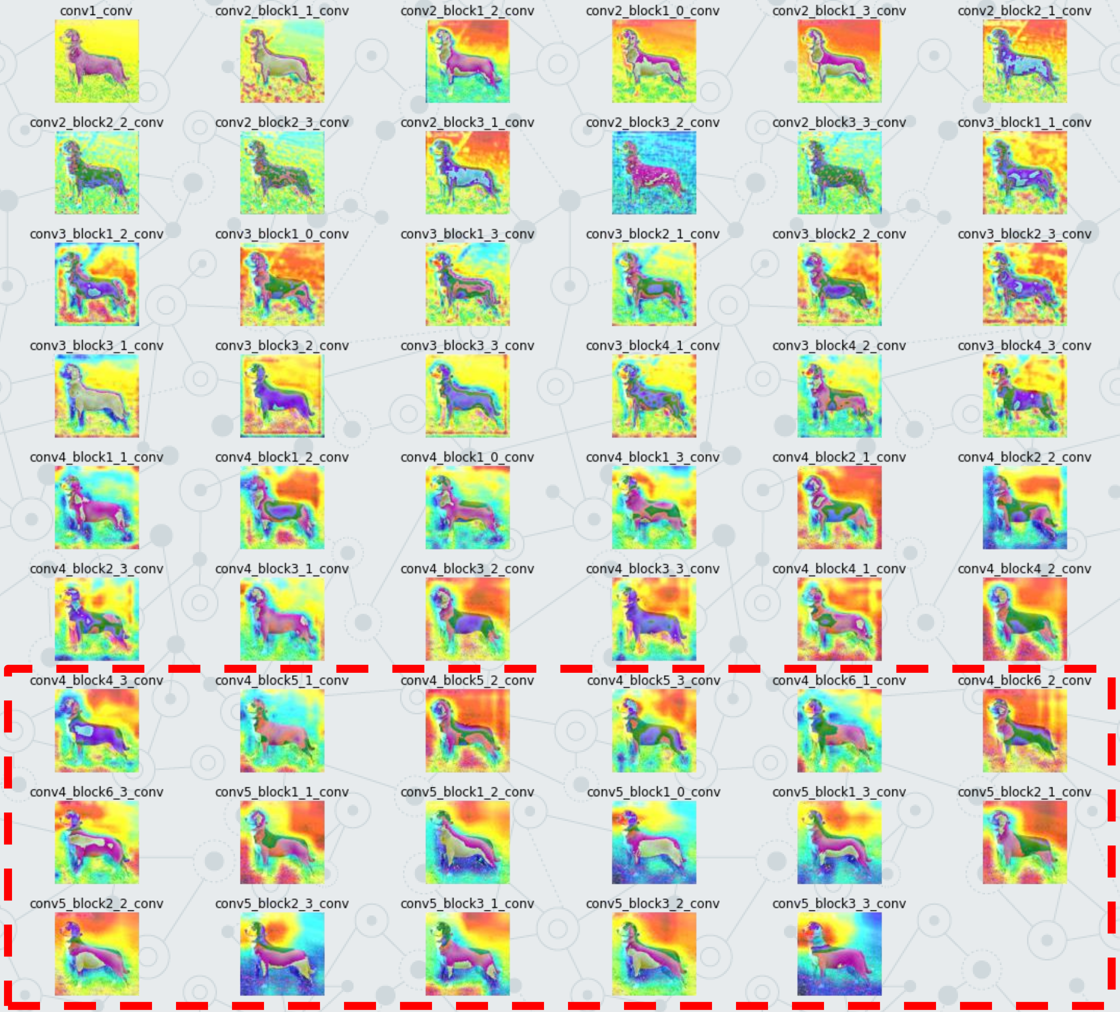
Original Image

conv4_block4_2_conv heat-map



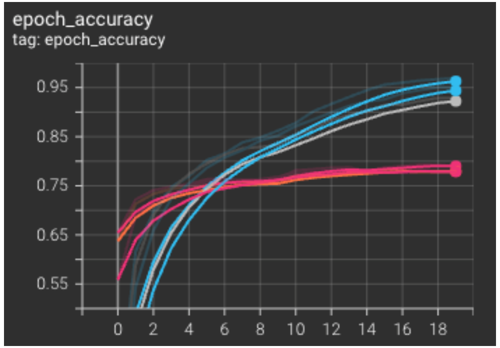
conv1_conv
conv2_block1_1_conv
conv2_block1_2_conv
conv2_block1_0_conv
conv2_block1_3_conv
conv2_block2_1_conv
conv2_block2_2_conv
conv2_block2_3_conv
conv2_block3_1_conv
conv2_block3_2_conv
conv2_block3_3_conv
conv3_block1_1_conv
conv3_block1_2_conv
conv3_block1_0_conv
conv3_block1_3_conv
conv3_block2_1_conv
conv3_block2_2_conv
conv3_block2_3_conv
conv3_block3_1_conv
conv3_block3_2_conv
conv3_block3_3_conv
conv3_block4_1_conv
conv3_block4_2_conv
conv3_block4_3_conv
conv4_block1_1_conv
conv4_block1_2_conv
conv4_block1_0_conv
conv4_block1_3_conv
conv4_block2_1_conv
conv4_block2_2_conv
conv4_block2_3_conv
conv4_block3_1_conv
conv4_block3_2_conv
conv4_block3_3_conv
conv4_block4_1_conv
conv4_block4_2_conv
conv4_block4_3_conv
conv4_block5_1_conv
conv4_block5_2_conv
conv4_block5_3_conv
conv4_block6_1_conv
conv4_block6_2_conv
conv4_block6_3_conv
conv5_block1_1_conv
conv5_block1_2_conv
conv5_block1_0_conv
conv5_block1_3_conv
conv5_block2_1_conv
conv5_block2_2_conv
conv5_block2_3_conv
conv5_block3_1_conv
conv5_block3_2_conv
conv5_block3_3_conv


conv1_conv
conv2_block1_1_conv
conv2_block1_2_conv
conv2_block1_0_conv
conv2_block1_3_conv
conv2_block2_1_conv
conv2_block2_2_conv
conv2_block2_3_conv
conv2_block3_1_conv
conv2_block3_2_conv
conv2_block3_3_conv
conv3_block1_1_conv
conv3_block1_2_conv
conv3_block1_0_conv
conv3_block1_3_conv
conv3_block2_1_conv
conv3_block2_2_conv
conv3_block2_3_conv
conv3_block3_1_conv
conv3_block3_2_conv
conv3_block3_3_conv
conv3_block4_1_conv
conv3_block4_2_conv
conv3_block4_3_conv
conv4_block1_1_conv
conv4_block1_2_conv
conv4_block1_0_conv
conv4_block1_3_conv
conv4_block2_1_conv
conv4_block2_2_conv
conv4_block2_3_conv
conv4_block3_1_conv
conv4_block3_2_conv
conv4_block3_3_conv
conv4_block4_1_conv
conv4_block4_2_conv
conv4_block4_3_conv
conv4_block5_1_conv
conv4_block5_2_conv
conv4_block5_3_conv
conv4_block6_1_conv
conv4_block6_2_conv
conv4_block6_3_conv
conv5_block1_1_conv
conv5_block1_2_conv
conv5_block1_0_conv
conv5_block1_3_conv
conv5_block2_1_conv
conv5_block2_2_conv
conv5_block2_3_conv
conv5_block3_1_conv
conv5_block3_2_conv
conv5_block3_3_conv


conv1_conv
conv2_block1_1_conv
conv2_block1_2_conv
conv2_block1_0_conv
conv2_block1_3_conv
conv2_block2_1_conv
conv2_block2_2_conv
conv2_block2_3_conv
conv2_block3_1_conv
conv2_block3_2_conv
conv2_block3_3_conv
conv3_block1_1_conv
conv3_block1_2_conv
conv3_block1_0_conv
conv3_block1_3_conv
conv3_block2_1_conv
conv3_block2_2_conv
conv3_block2_3_conv
conv3_block3_1_conv
conv3_block3_2_conv
conv3_block3_3_conv
conv3_block4_1_conv
conv3_block4_2_conv
conv3_block4_3_conv
conv4_block1_1_conv
conv4_block1_2_conv
conv4_block1_0_conv
conv4_block1_3_conv
conv4_block2_1_conv
conv4_block2_2_conv
conv4_block2_3_conv
conv4_block3_1_conv
conv4_block3_2_conv
conv4_block3_3_conv
conv4_block4_1_conv
conv4_block4_2_conv
conv4_block4_3_conv
conv4_block5_1_conv
conv4_block5_2_conv
conv4_block5_3_conv
conv4_block6_1_conv
conv4_block6_2_conv
conv4_block6_3_conv
conv5_block1_1_conv
conv5_block1_2_conv
conv5_block1_0_conv
conv5_block1_3_conv
conv5_block2_1_conv
conv5_block2_2_conv
conv5_block2_3_conv
conv5_block3_1_conv
conv5_block3_2_conv
conv5_block3_3_conv

Challenges

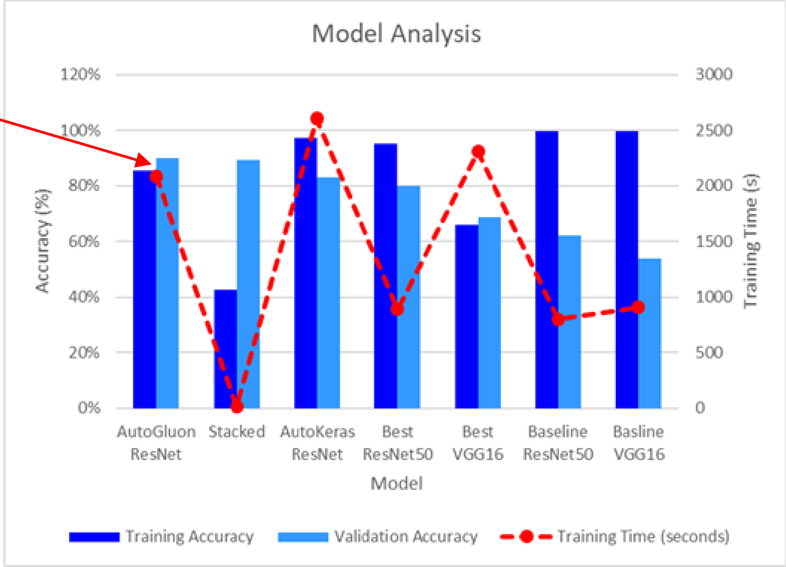
- Goal when training was to optimize the accuracy on the validation set
- Encountered both underfitting and overfitting using transfer learning
- Strategies to tackle underfitting:
  - Adding a dense layer followed by a dropout prior to the activation layer
- Strategies to tackle overfitting:
  - Layer manipulation – freezing and unfreezing convolutional layers in the pre-trained model
  - Global pooling layers
  - Batch normalization layers



Summary

Transfer learning, ensemble, and AutoML models were used to build multiclass image classifiers for a dataset of different dog breed images containing 120 different classes

AutoGluon had a val. accuracy of 89.9% and a test accuracy of 91% on test set



Thank You!

Any questions?

Brandon Attai
Tahsin Chowdhury
Kelten Falez
Timothy Mok
Aron Saengchan
Yong Jun Zhu