Shoppe - Price Match Guarantee

Agenda

Part 1: Problem Statement
Part 2: Data
Part 3: Methodology
- CNN
- ArcFace
- ArcFace + TD-IDF
- ArcFace + BERT

Part 4: Results & Future Work

Part 1:

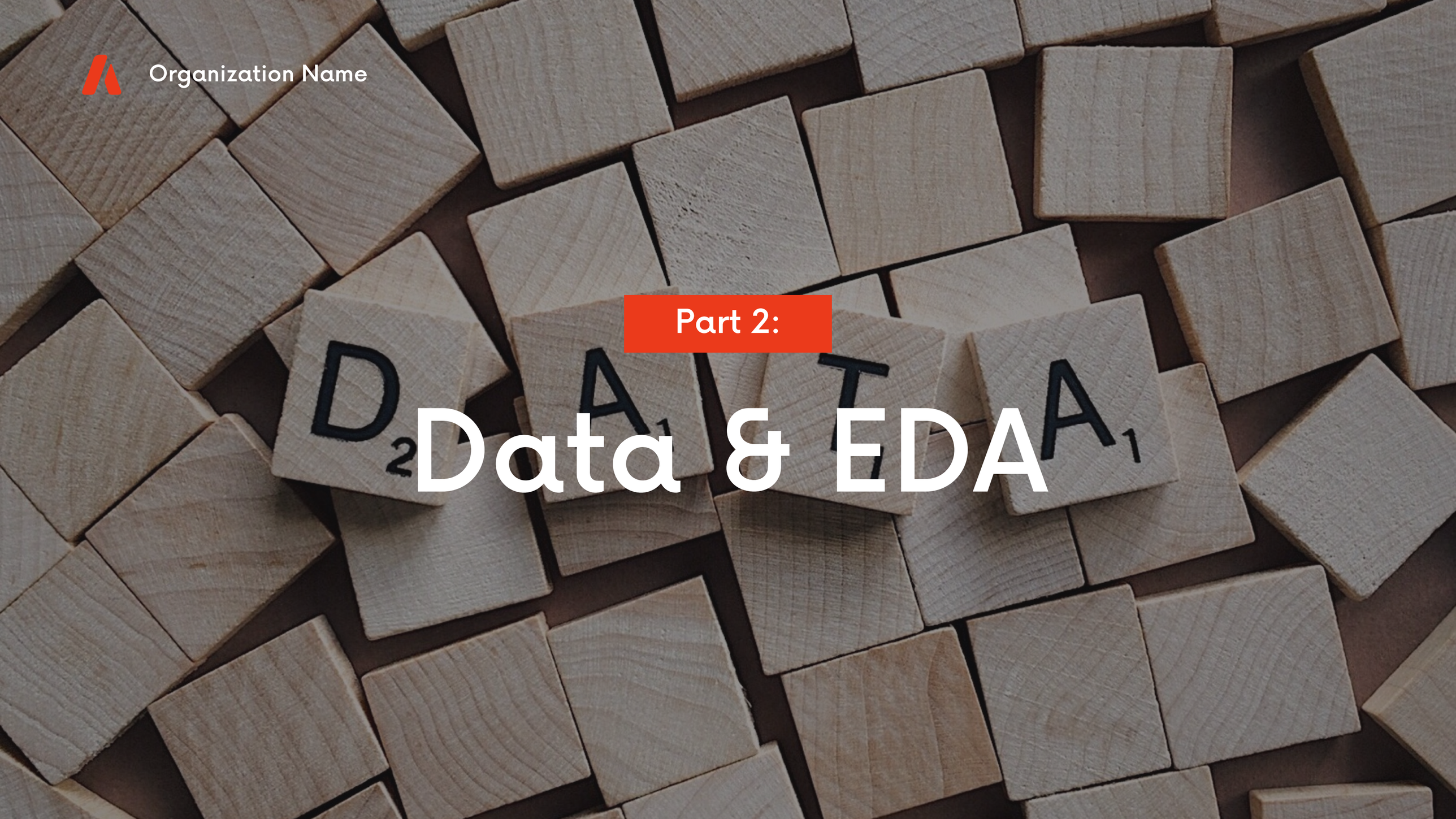
Problem Statement

Problem Statement

Retail companies use a variety of methods to assure customers that their products are the cheapest. Among them is product matching, which allows a company to offer products at rates that are competitive to the same product sold by another retailer.

Two different images of similar wares may represent the same product or two completely different items. Retailers want to avoid misrepresentations and other issues that could come from conflating two dissimilar products. Currently, a combination of deep learning and traditional machine learning analyzes image and text information to compare similarity. But major differences in images, titles, and product descriptions prevent these methods from being entirely effective.

To solve this problem, our team built several Deep Learning models that predict which items are the same products based on the 34520 images provided.

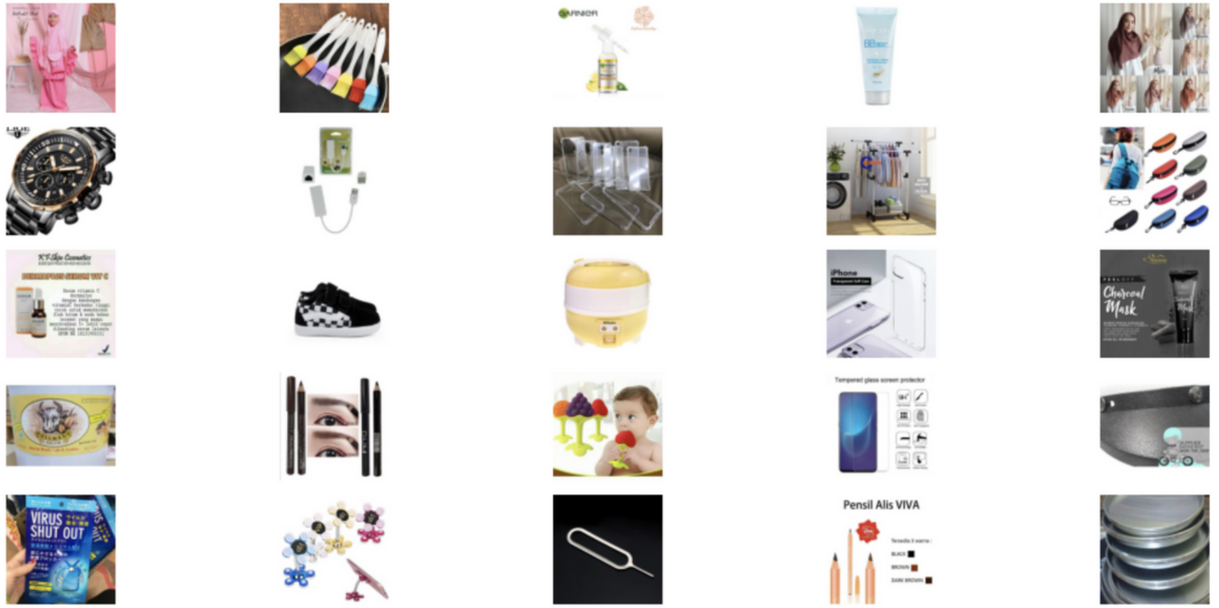

Organization Name
Part 2:
Data & EDA

Explorative Data Analysis

Images:

-Training size: 34520

- Testing Size**:** 3
- No missing fields
- Most products have 5 images or fewer
- The most data-rich products have 51 images



Labels:

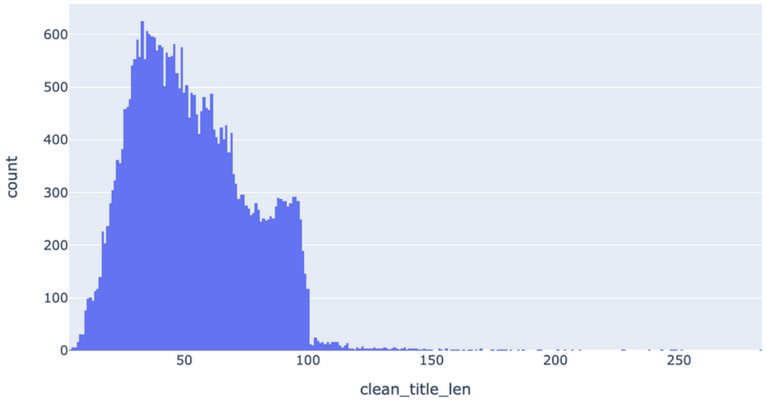
- **posting_id:** the ID code for the posting.
- **image:** the image id/md5sum.
- **image_phash:** a perceptual hash of the image.
- **title:** the product description for the posting.
- **label_group:** ID code for all postings that map to the same product.

Explorative Data Analysis

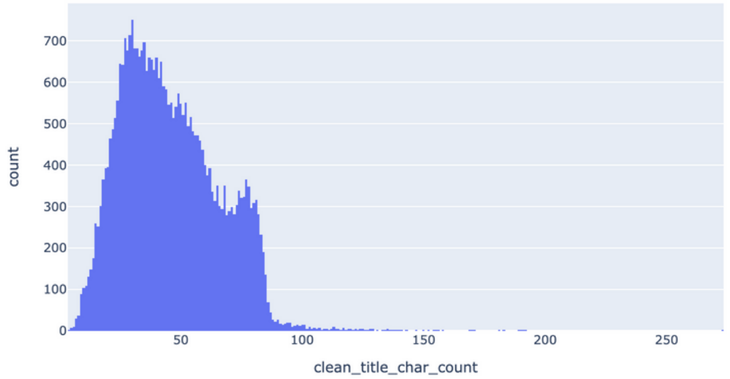
Label

11014 Unique Labels

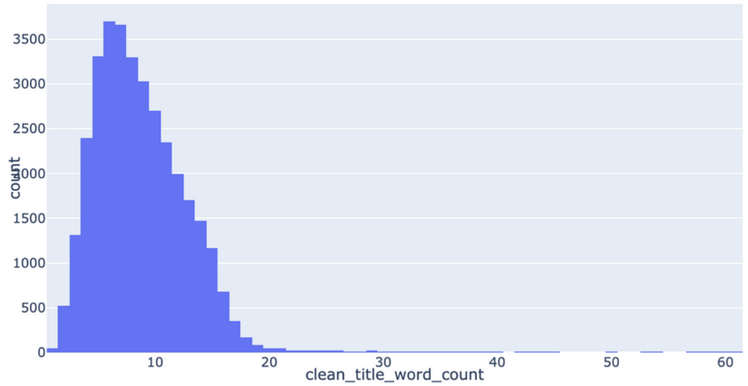
Title Length Distribution

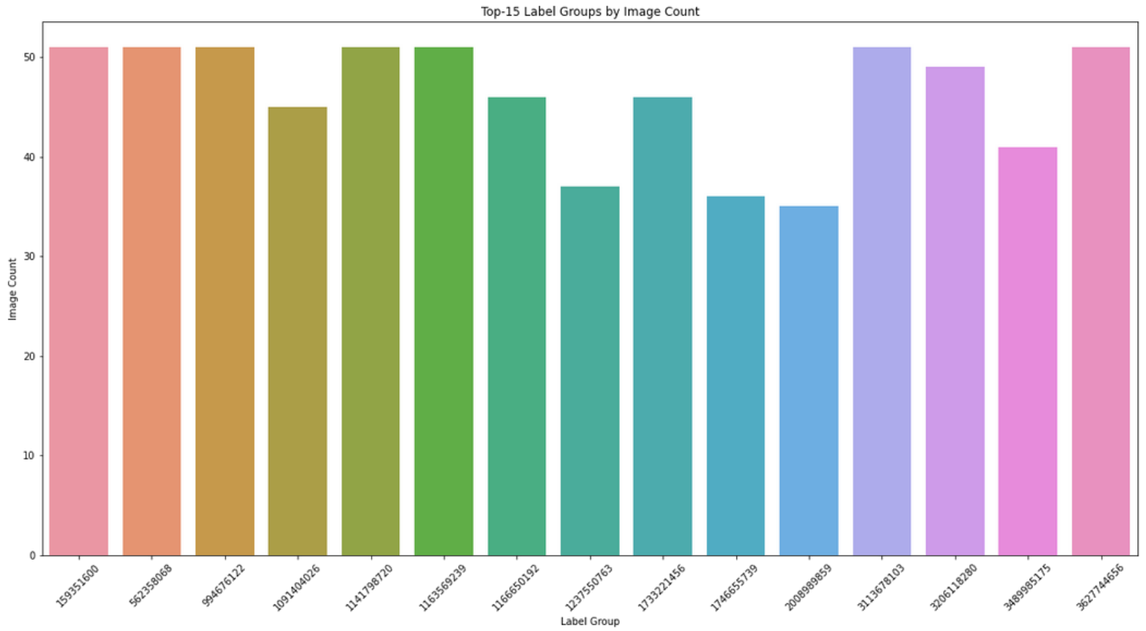


Character Count Distribution



Word Count Distribution





Explorative Data Analysis

Image Exploration

1. **Image Dimension**

(640, 640,3)

2. **Image Byte Scale**

(0,255)

3. **Matrix of Pixels by Color**

(40960， 3)

4. **Training Data Shape**

(34250, 5)

5. **Test Dataset Shape**

(3, 4)

Part 3:

Methodology

What is our goal?

Idea: Find all listings that belong to the same product.

Deliverable: for each posting_id, predict all posting_ids that belong to the same label_group.

Image classification does not work very well because

1. Very limited images per label_group
2. Does not adapt well to new products

What do we do?

1. Train an image classification model to learn features from images
2. **Embed** images using the trained classification model
3. Use **similarity matching** to find similar products (predicting label_group is no longer the goal)
4. Find the optimal threshold for nearest neighbor

Evaluation Metric - f1

- Evaluation Metric: f1 = $\frac{2*num\_correct\_prediction}{num\_y\_true+num\_y\_pred}$
- An f1 score is calculated for every predicted row, then averaged to get the final f1 score.

Image Classifier 1: Basic CNN

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
efficientnetb5 (Functional)  (None, None, None, 2048)  28513527
_________________________________________________________________
global_average_pooling2d (Gl (None, 2048)              0
_________________________________________________________________
batch_normalization (BatchNo (None, 2048)              8192
_________________________________________________________________
dense (Dense)                (None, 1024)              2098176
_________________________________________________________________
batch_normalization_1 (Batch (None, 1024)              4096
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_3 (Dense)              (None, 11014)             11289350
=================================================================
Total params: 44,012,541
Trainable params: 43,833,654
Non-trainable params: 178,887
_________________________________________________________________
```

Setup

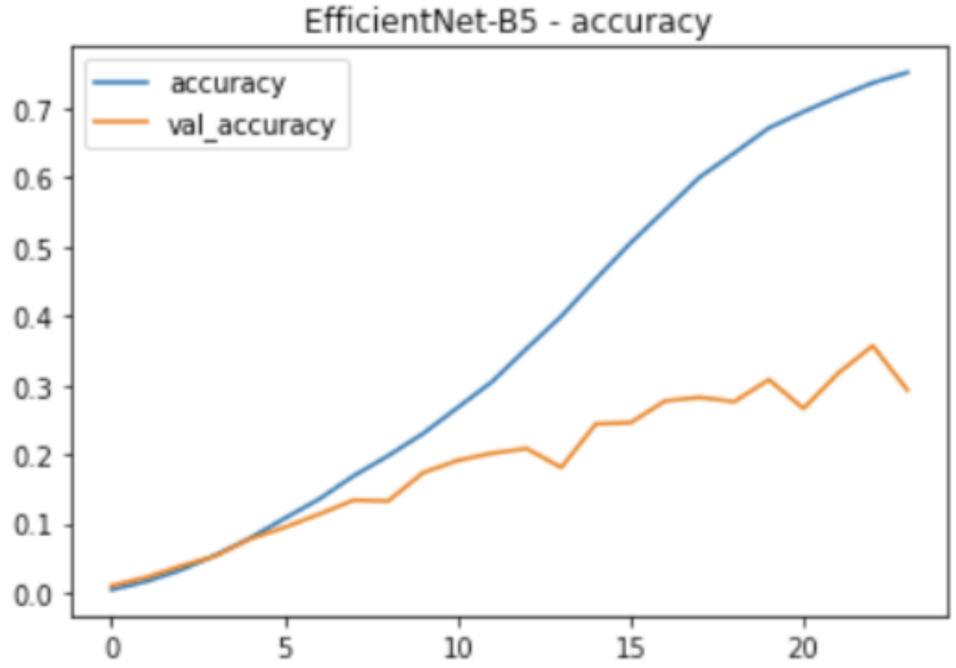
- predictor: image
- target: label_group
- data augmented: flip, adjust hue, saturation, contrast & brightness
- batch_size = 8
- img_size = (224,224)
- Scheduler: Exponential decay scheduler(0.0001, decay_rate=1e-6)
- Optimizer: Adam
- EfficientNet-B5 (weights='imagenet')
- Trained for 24 epochs: stopped by EarlyStopping

Results

- training loss: 0.9294
- training accuracy: 0.7511
- val_loss: 13.3457
- val_accuracy: 0.2927
- f1: 0.1523

Image Classifier 1: Basic CNN

- Training accuracy keeps climbing
- Validation loss keeps climbing
- Validation loss and accuracy both increase
  - The model is making more accurate predictions, but it's becoming overconfident on the wrongly predicted images, hence the increasing loss
  - Could grow worse if we train for more epochs



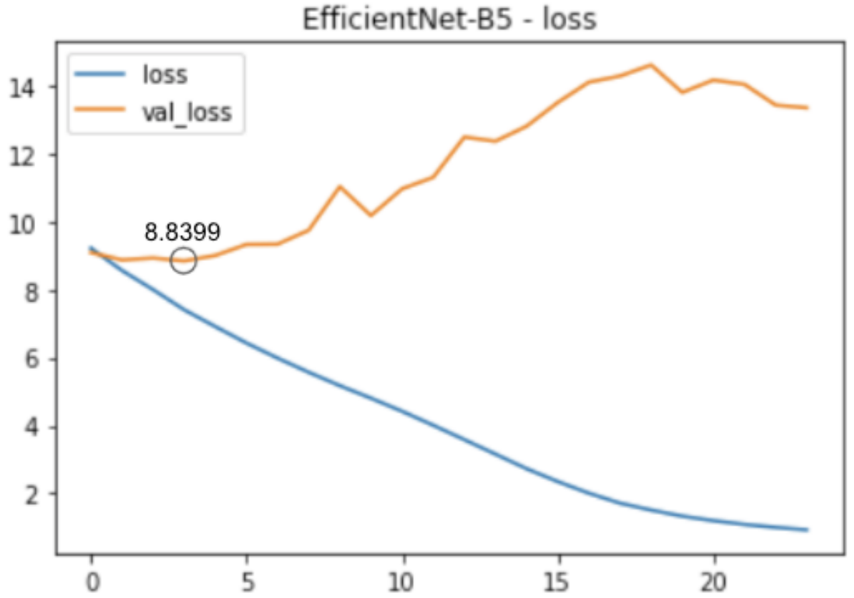


Image Classifier 2: ArcFace

Softmax $$L_1 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_i + b_j}}$$

ArcFace $$L_3 = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{e^{s(\cos(\theta_{y_i}+m))}}{e^{s(\cos(\theta_{y_i}+m))} + \sum_{j=1, j\neq y_i}^{n} e^{s\cos\theta_j}}$$

- What is ArcFace? - A **loss function** transformed from the Softmax loss function
- Learns highly discriminative features for robust face recognition
- ArcFace adds more loss to **encourage similar class embeddings to be close and dissimilar embeddings to be far from each other**. This adds a second task to the training of a classifier. (The first task is predicting the image accurately) - **perfect for similarity matching**
- More stable than previous versions of angular margin based loss functions (e.g. SphereFace, CosFace)
- Outperforms STOA face recognition tasks consistently
- Adds negligible computational overhead

Image Classifier 2: ArcFace

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inp1 (InputLayer)            [(None, 512, 512, 3)      0
_________________________________________________________________
efficientnetb3 (Functional)  (None, None, None, 1      10783535
_________________________________________________________________
global_average_pooling2d (Globa (None, 1536)           0
_________________________________________________________________
inp2 (InputLayer)            [(None,)]                 0
_________________________________________________________________
head/arc_margin (ArcMarginProdu (None, 11014)          16917504

_________________________________________________________________
softmax (Softmax)            (None, 11014)             0
=================================================================
Total params: 27,701,039
Trainable params: 27,613,736
Non-trainable params: 87,303
_________________________________________________________________
```

Setup

- predictor: image
- target: label_group
- data augmented: flip, adjust hue, saturation, contrast & brightness
- batch_size = 8
- img_size = (512, 512)
- Scheduler: custom scheduler, stepwise warmup and decay
- Optimizer: Adam, lr=1e-6
- Pretrained model: EfficientNet-B3 (weights='imagenet')
- Trained for 56 epochs (limited time)
- It has not converged; val_loss is still declining - room for improvement

Results

- training loss: 0.8803
- training accuracy: 0.9208
- val_loss: 8.7221
- val_accuracy: 0.4604
- f1: 0.9144

How do we further improve matching? -> text embedding

Text Embedding 1: TF-IDF+KNN

TF-IDF:

Calculate the weight factors of words in the corpus to find the important words

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

- predictor: image title
- target: label_group

KNN:

Calculate word similarity to match sentences

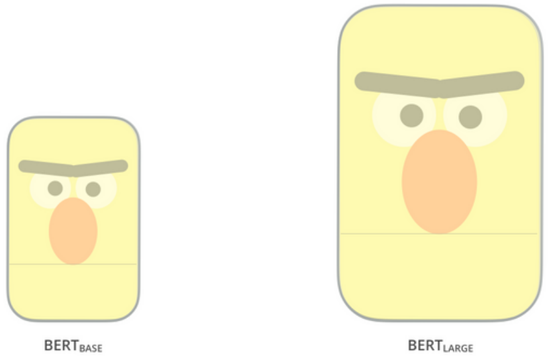
Results:

- threshold: 0. 44
- F1 score: 0.67

Text Classifier 2: BERT

BERT:
- The pre-trained unsupervised natural language processing model stands for Bidirectional Encoder Representations from Transformers.

- BERT Base with 12 layers and 768 hides was used in this project



- predictor: image title
- target: label_group
- batch_size = 32
- Optimizer: Adam
- Loss Function: ArcFace
- Trained for 25 epochs

Results

- training loss: 0.8604
- train accuracy: 0.9827
- val_loss: 15.25
- val_accuracy: 0.1662

Similarity Matching

Nearest neighbor search (cosine similarity) based on:

1. Image embedding:
    a. Basic CNN
    b. ArcFace
2. Text embedding:
    a. tfidf
    b. Bert
3. Mixed:
    a. ArcFace+tfidf

Results

- Image
  - Basic CNN: best f1 = 0.1523; threshold = 0.2
  - **ArcFace: best f1 = 0.9144; threshold = 0.54**
- Text
  - tfidf: best f1 = 0.6661; threshold = 0.44
- Combined
  - ArcFace+tfidf: best f1 =0.8398

Notes
- These scores are based on the full dataset.
- The validation set was left out during training and added during evaluation.
- Test set has 70k images (private dataset), similar approaches show ~.70 f1 score.

Results & Future Work

Results & Future Work

1. Combine Bert (Text) Model and EfficientNet-B3 model (Image Model) with ArcFace to generate the final classification
2. Add more image preprocessing to reduce the overfitting and improve validation performance
3. Fine tuning the hyperparameters to improve the ArcFace model performance

Thank you!