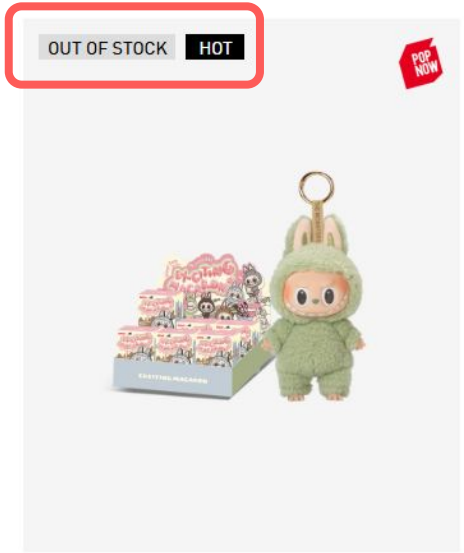
Detecting Fake vs. Real Dolls (Labubus)

Joseph Bae
DTSA 5511 Final Project
July 2025

The Labubu Craze

Trying to buy a Labubu is hard

Stores are often out of stock



THE MONSTERS

THE MONSTERS - Exciting Macaron Vinyl Face Blind Box

$27.99

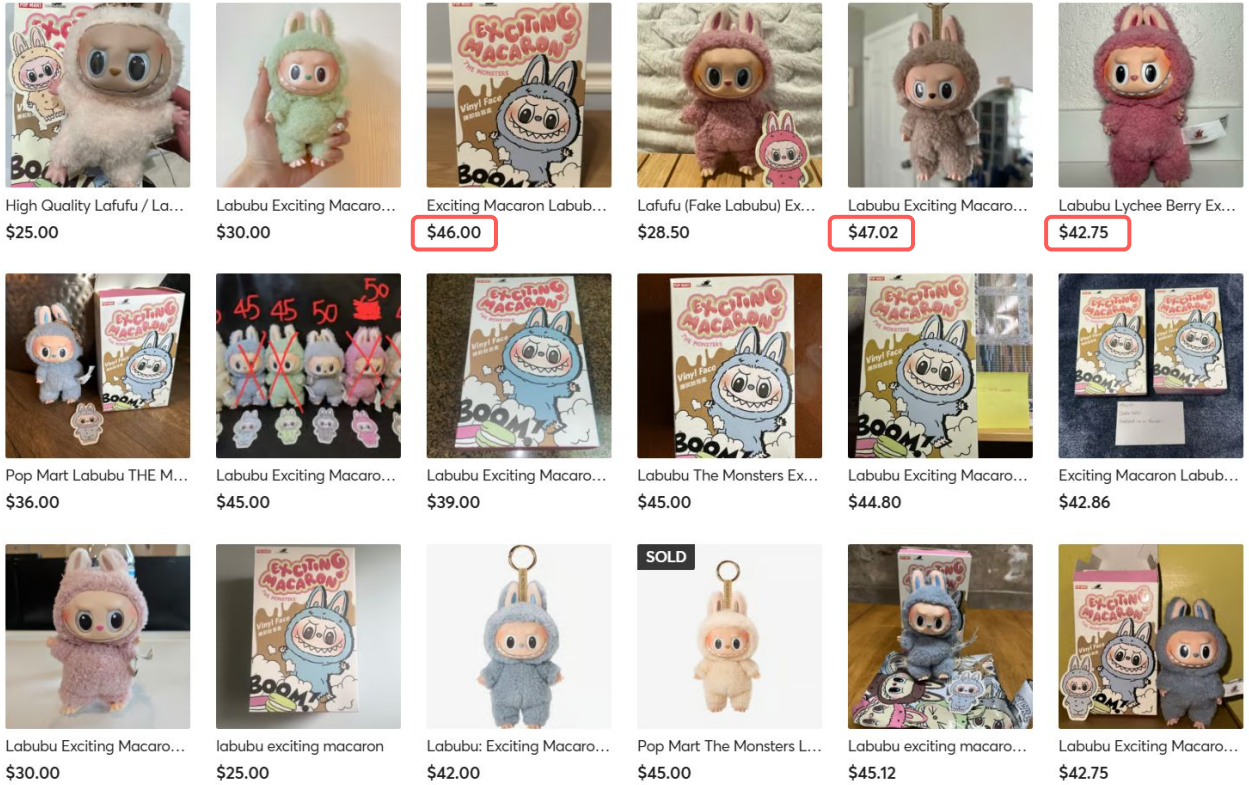
We are currently out of stock

of ALL labubus and we have not received any of the new energy series yet - we have no idea when we will next receive a restock and we will post on our socials when we receive any - our DMs are currently flooded with questions and unfortunately we don't have the time to reply to all of them

Buying Labubus Online

Original price from manufacturer is $27.99

Resellers sell for $25 to $45+



Fake Postings

Labubu For Sale

$30

~~$40~~

Last updated 1 day ago in

Condition: Used (normal wear)

Toys, Games, & Hobbies - Stuffed animals & Plush - Plush figures

Make offer

Description

Opened Labubu for sale. My daughter had to have it, but now she doesn't want it. It is opened, no box, no bag. It is as is. Doll was hard to get. EDIT: Price lowered as recommended

Fake Postings

labubu

$200

Posted 14 days ago in

Condition: New

Toys, Games, & Hobbies - Toys - Other - Toys

Make offer

Description

from pop mart if you would like more pictures let me know! and i am located in

A DL model that can detect fakes

REAL

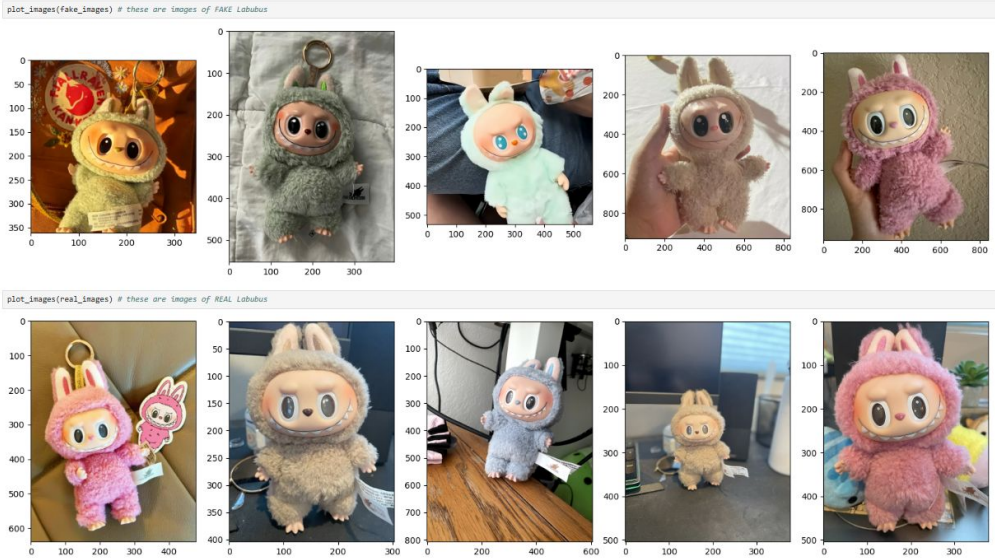
- Reassure consumer
- Allow posting on marketplace

FAKE

- Warn consumer
- Block from posting on marketplace

Dataset

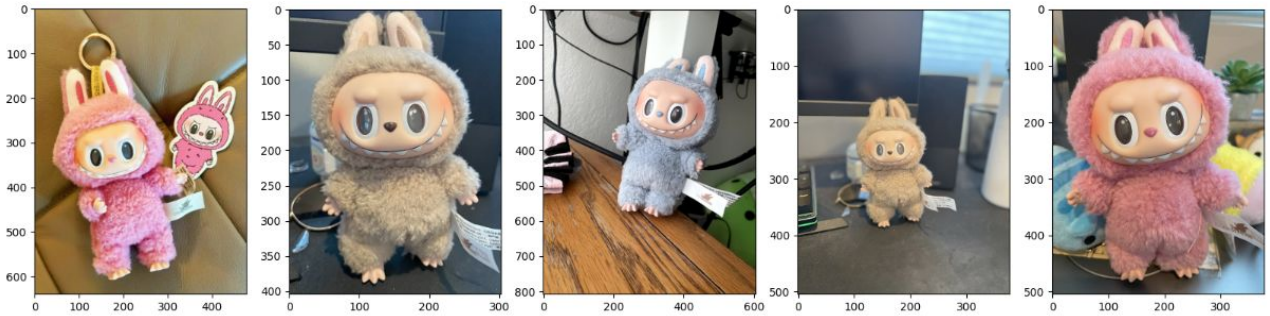
- 200 images of Labubus (100 real, 100 fake)
- 60/20/20 training/validation/test split



Real vs. Fake Labubus

Real Labubus

- Oval face shape
- Matte face texture
- No visible stitching



Fake Labubus

- Varying face shape
- Varying pupil sizes
- Glossy face texture
- Visible stitching

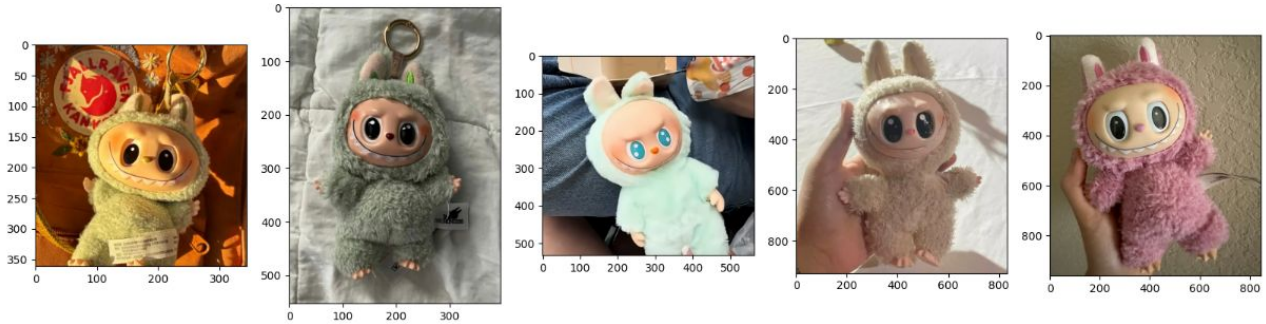
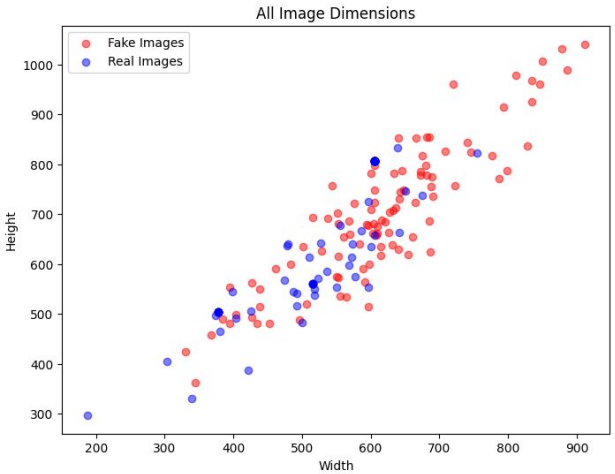
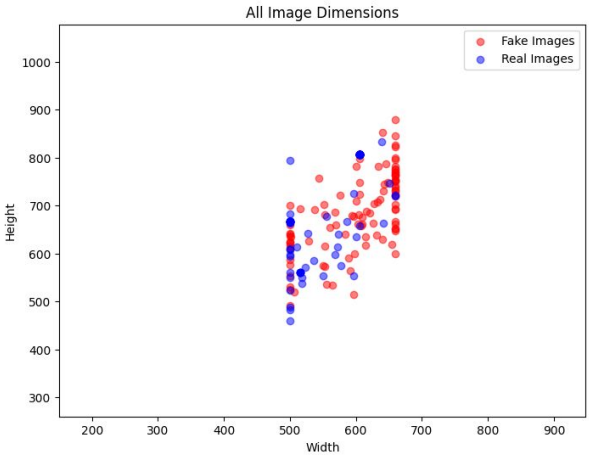


Image Resizing

Keras CNN models require image inputs to be the same size. Image resizing was done in 2 steps:



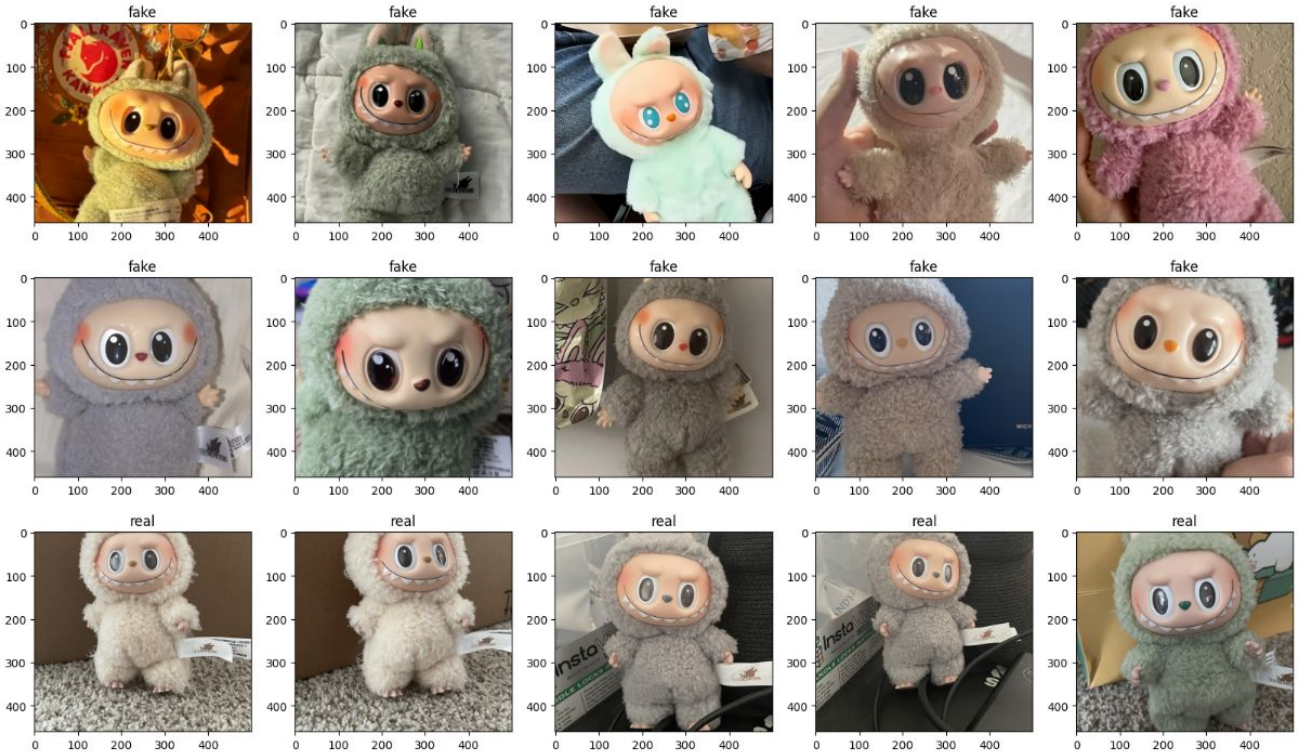
1. CV2 Image Resizer



2. Cropping to smallest image's size (500x460)

Image Resizing

Images are now the same size, but some are cut off



Other Data Processing

- Standardize image data (subtract mean, divide by std. dev)
- Split into 60/20/20 training/validation/test
- Shuffle the training images so they’re fed into Keras model randomly

Model Building

Model 1	3 convolutional layers > max pooling > dense > binary classification	129M parameters
Model 2	additional max pooling placed between all conv. layers	7.6M parameters
Model 3	conv. layers have stride 2	98K parameters
Model 4	use image augmentation: random flip and rotation	98K parameters
Model 5	same as model 3 but with 50% dropout after dense layer	98K parameters
Model 6	one additional dense layer and another 50% dropout before output	98K parameters

Model Training

- Training images: 120
- Validation images: 40

- Binary classification:
  - Real = 0, Fake = 1

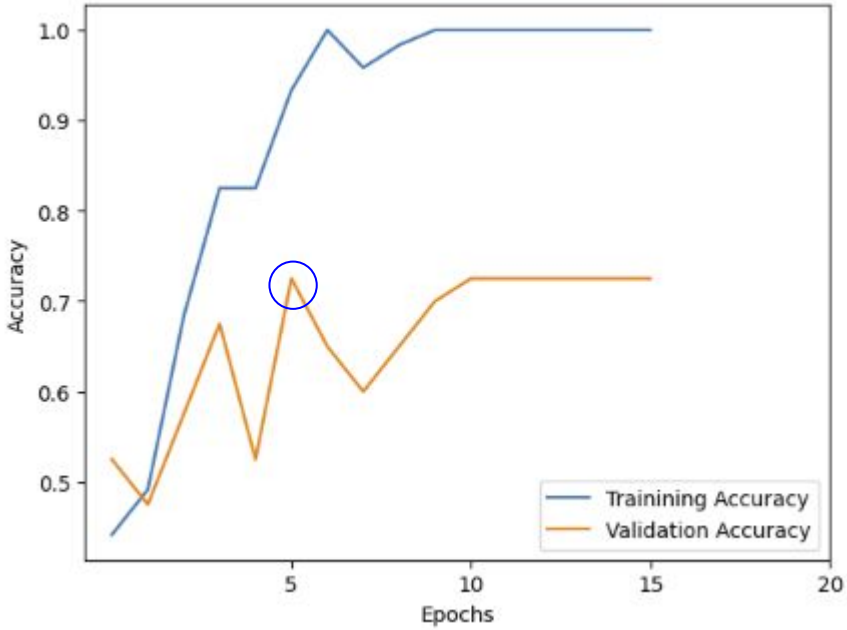
- Callbacks:
  - EarlyStopping - stops training when validation accuracy fails to increase for 10 epochs
  - ReduceLROnPlateau - lowers learning rate when validation loss fails to increase for 10 epochs

Model Performance

Model 1

- Most parameters (129M)
- Overfitting on training data
- Validation accuracy: 73%

```
model_v1 = Sequential()
model_v1.add(Conv2D(filters = 3, kernel_size = (3, 3), activation = 'relu'))
model_v1.add(Conv2D(filters = 6, kernel_size = (3, 3), activation = 'relu'))
model_v1.add(Conv2D(filters = 9, kernel_size = (3, 3), activation = 'relu'))
model_v1.add(MaxPooling2D(strides = 2))
model_v1.add(Flatten())
model_v1.add(Dense(units = 256, activation = 'relu'))
model_v1.add(Dense(units = 1, activation = 'sigmoid'))
```

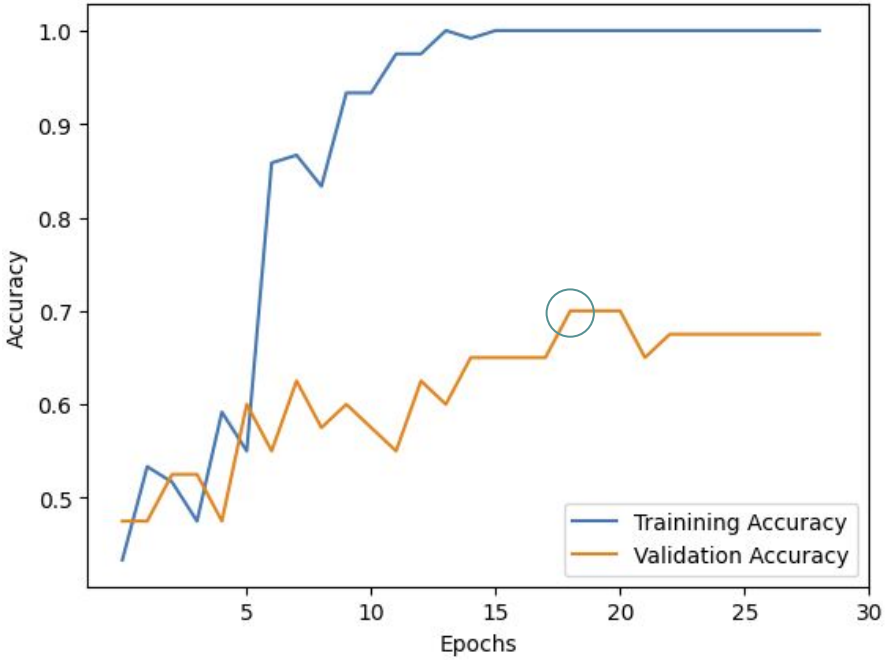


Model Performance

Model 2

- Many parameters (7.6M)
- Overfitting, but took more epochs
- Validation accuracy: 70%

```
model_v2 = Sequential()
model_v2.add(Conv2D(filters = 3, kernel_size = (3, 3), activation = 'relu'))
model_v2.add(MaxPooling2D(strides = 2))
model_v2.add(Conv2D(filters = 6, kernel_size = (3, 3), activation = 'relu'))
model_v2.add(MaxPooling2D(strides = 2))
model_v2.add(Conv2D(filters = 9, kernel_size = (3, 3), activation = 'relu'))
model_v2.add(MaxPooling2D(strides = 2))
model_v2.add(Flatten())
model_v2.add(Dense(units = 256, activation = 'relu'))
model_v2.add(Dense(units = 1, activation = 'sigmoid'))
```

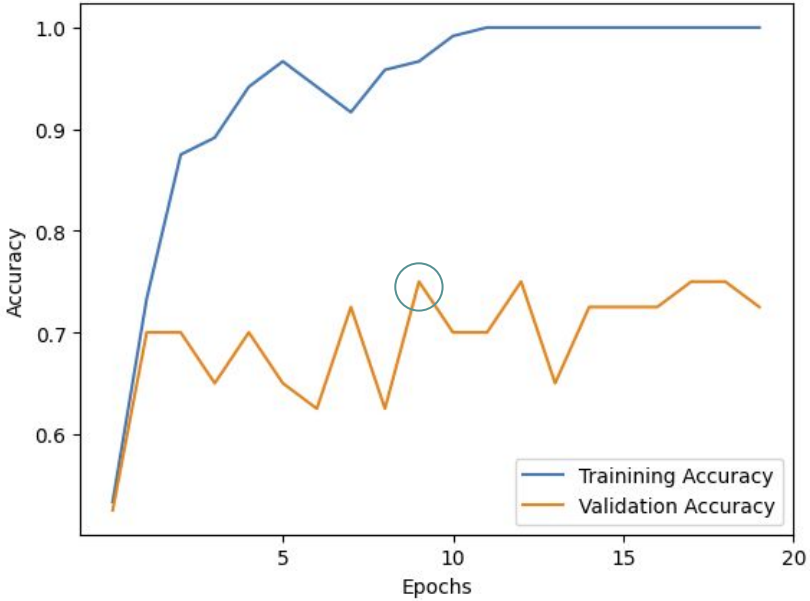


Model Performance

Model 3

- Less Parameters (98K)
- Overfitting
- Validation accuracy: 75%

```
model_v3 = Sequential()
model_v3.add(Conv2D(filters = 3, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v3.add(MaxPooling2D(strides = 2))
model_v3.add(Conv2D(filters = 6, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v3.add(MaxPooling2D(strides = 2))
model_v3.add(Conv2D(filters = 9, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v3.add(MaxPooling2D(strides = 2))
model_v3.add(Flatten())
model_v3.add(Dense(units = 256, activation = 'relu'))
model_v3.add(Dense(units = 1, activation = 'sigmoid'))
```

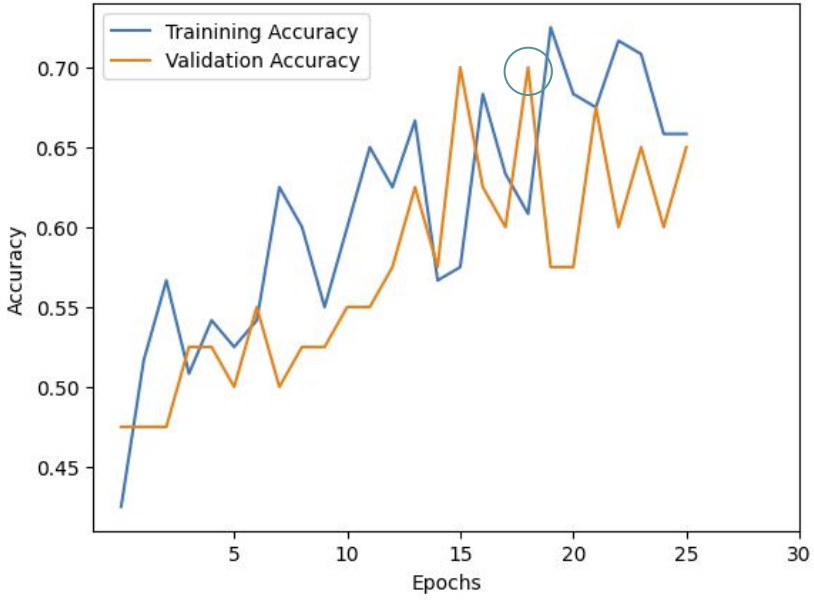


Model Performance

Model 4 (use Image Augmentation)

- Less Parameters (98K)
- No overfitting, but unstable accuracy
- Validation accuracy: 70%

```
model_v4 = Sequential()
model_v4.add(RandomFlip(mode = "horizontal"))
model_v4.add(RandomRotation(factor = 0.2, fill_mode = 'reflect'))
model_v4.add(Conv2D(filters = 3, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v4.add(MaxPooling2D(strides = 2))
model_v4.add(Conv2D(filters = 6, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v4.add(MaxPooling2D(strides = 2))
model_v4.add(Conv2D(filters = 9, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v4.add(MaxPooling2D(strides = 2))
model_v4.add(Flatten())
model_v4.add(Dense(units = 256, activation = 'relu'))
model_v4.add(Dense(units = 1, activation = 'sigmoid'))
```

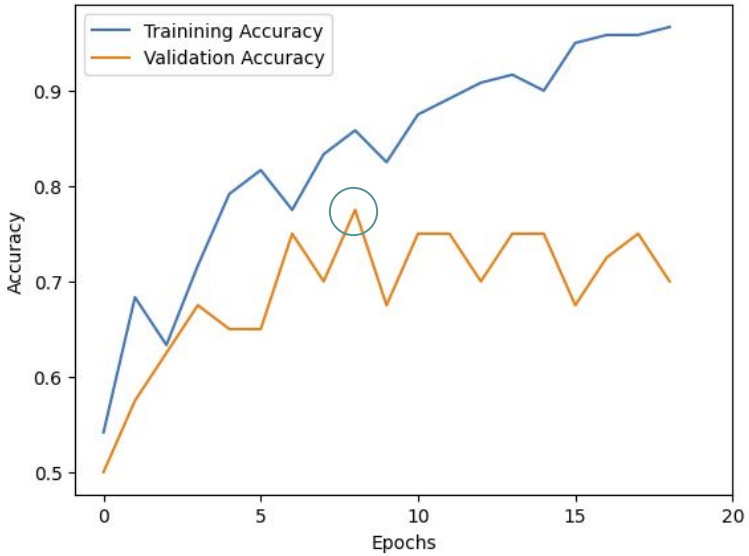


Model Performance

Model 5

- Less Parameters (98K)
- Less overfitting issue
- Validation accuracy: 78%

```
model_v5 = Sequential()
model_v5.add(Conv2D(filters = 3, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v5.add(MaxPooling2D(strides = 2))
model_v5.add(Conv2D(filters = 6, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v5.add(MaxPooling2D(strides = 2))
model_v5.add(Conv2D(filters = 9, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v5.add(MaxPooling2D(strides = 2))
model_v5.add(Flatten())
model_v5.add(Dense(units = 256, activation = 'relu'))
model_v5.add(Dropout(rate = 0.5))
model_v5.add(Dense(units = 1, activation = 'sigmoid'))
```

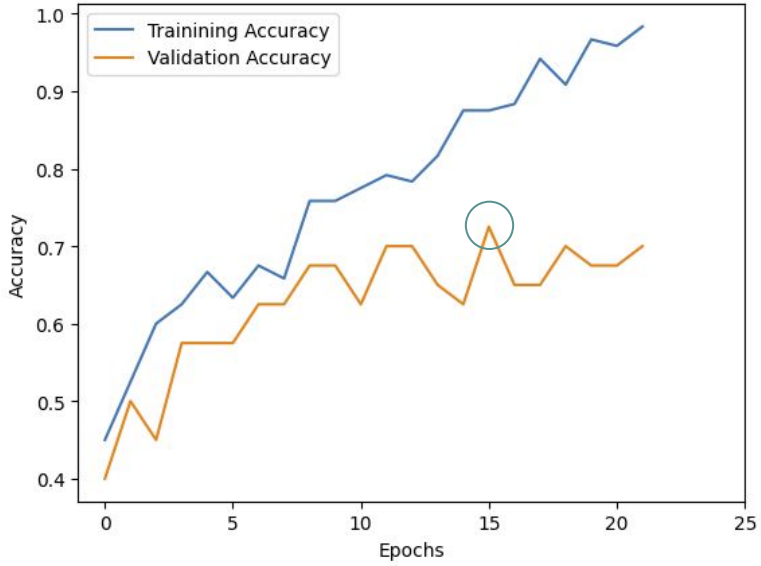


Model Performance

Model 6

- Less Parameters (98K)
- Similar trend as Model 5
- Validation accuracy: 73%

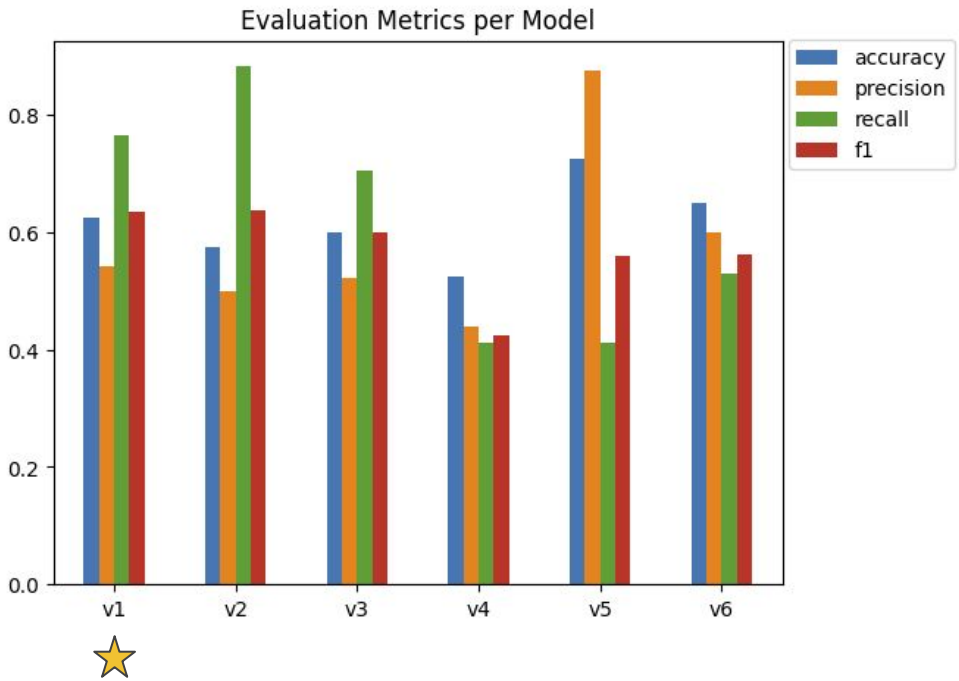
```
model_v6 = Sequential()
model_v6.add(Conv2D(filters = 3, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v6.add(MaxPooling2D(strides = 2))
model_v6.add(Conv2D(filters = 6, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v6.add(MaxPooling2D(strides = 2))
model_v6.add(Conv2D(filters = 9, kernel_size = (3, 3), strides = 2, activation = 'relu'))
model_v6.add(MaxPooling2D(strides = 2))
model_v6.add(Flatten())
model_v6.add(Dense(units = 256, activation = 'relu'))
model_v6.add(Dropout(rate = 0.5))
model_v6.add(Dense(units = 128, activation = 'relu'))
model_v6.add(Dropout(rate = 0.5))
model_v6.add(Dense(units = 1, activation = 'sigmoid'))
```



Model Performance

Model	Description	Training, Validation Accuracy
Model 1	3 convolutional layers > max pooling > dense > binary classification	100%, 73%
Model 2	additional max pooling placed between all conv. layers	100%, 70%
Model 3	conv. layers have stride 2	100%, 75%
Model 4	use image augmentation: random flip and rotation	61%, 70%
Model 5	same as model 3 but with 50% dropout after dense layer	87%, 78%
Model 6	one additional dense layer and another 50% dropout before output	87%, 73%

Model Performance on Test Data



- Model 5 had 73% accuracy, but very low recall, which is important for a fake detection model
- Model 1 had a good balance of metrics with high recall (77%)
- Model 2 had very high recall (88%) but fairly low accuracy (58%)
- Precision is relatively unimportant for the model, since false positives (real classified as fake) is not as dangerous for consumers as false negatives (fakes classified as real).

Conclusions

Model Findings

- Importance of a separate test dataset, which led to different model selection than just using training + validation scores
- How easy it is for DL models to overfit, esp. when training dataset is not large
- Fluctuation in accuracy metrics during training

Data Collection/Processing

- Handling images of different formats (PNG, JPEG, HEIC)
- Resizing images optimally, cropping around the key image object
    - Need a separately trained object detection model

Model Enhancements

- More images for better training, and to handle other types of Labubus.
- Use of pre-trained models which can perform well with small training data, since all it needs is fine-tuning/adjustments.

Model Challenges

- Very good quality fakes will be hard to detect
- Quality of genuine Labubus can also vary
- Limited to visual differences (e.g. textural differences could look identical)

Thank You