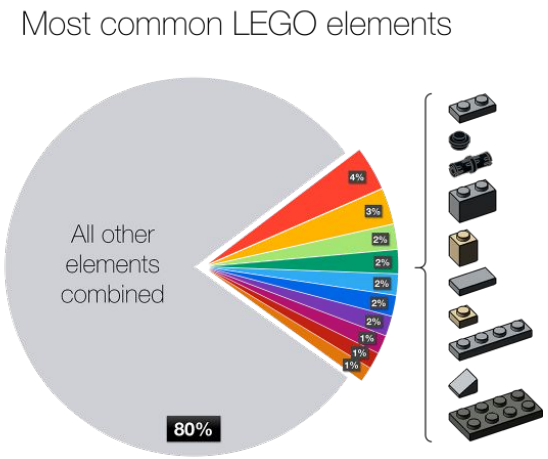
Classification of LEGO Bricks Based on Images

Daniel Mansfeldt, Tabea Rahm, Jule Kuhn

Introduction

- 3700+ unique LEGO bricks
- 10 most common bricks ≙ 20% of all elements

https://absatzwirtschaft-1cf93.kxcdn.com/wp-content/uploads/2022/08/lego-1024x676.jpg



https://brickarchitect.com/bricks/

- over 600 billion parts in the world
- 74 different brands of building blocks systems compatible with LEGO
- most bricks made from ABS (Acrylonitrile Butadiene Styrene)
  - ➔ cannot be recycled

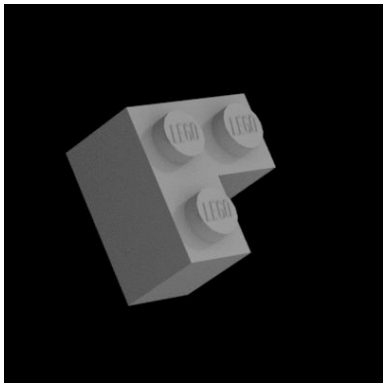
Data

- dataset:

  Images of LEGO Bricks

  40,000 images of 50 different LEGO bricks

  https://www.kaggle.com/datasets/joosthazelzet/lego-brick-images



- 50 bricks computer rendered by 800 different angles
- 2 camera setup
- 400 x 400 pixels, 3 RGB channels

Used Methods/Models - Preprocessing

- load pictures into pandas dataframe, get labels from file names

```
filenames = os.listdir(path)
filenames.remove('.gitkeep')
df = pd.DataFrame(filenames, columns=['Filenames'])

df['Label'] = df['Filenames'].apply(lambda x: get_label(x))
df = df.sample(frac=1,random_state=1).reset_index()
```

- get train and validation generator using flow_from_dataframe

```
train_generator = datagen.flow_from_dataframe(df,
                                              directory=path,
                                              x_col='Filenames',
                                              y_col='Label',
                                              target_size=image_size,
                                              batch_size=batch_size,
                                              color_mode='grayscale',
                                              class_mode='categorical',
                                              subset='training',
                                              )
```

Used Methods/Models - Tuner

- Setting up a Keras Tuner

```
tuner = keras_tuner.RandomSearch(
    hypermodel=build_model,
    objective="val_accuracy",
    max_trials=10,
    executions_per_trial=1,
    overwrite=True,
    directory="../experiments",
    project_name="Hyperparametertuning",
)
```

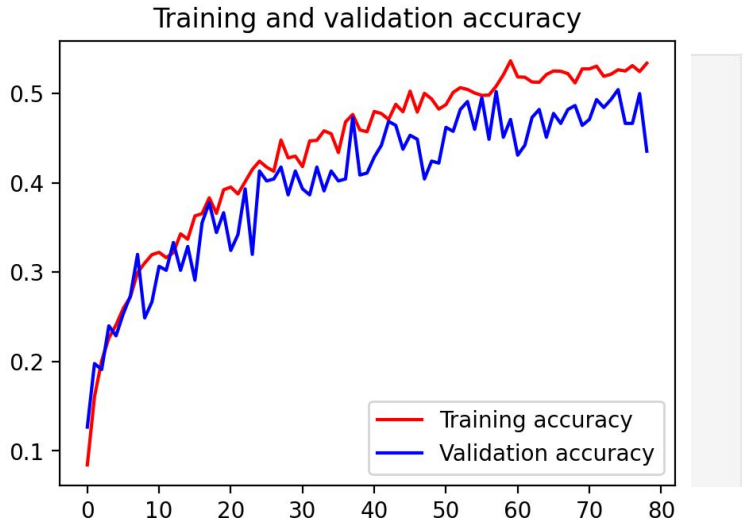
- Hyperparameter Grid based on Baseline Model
- Retraining of best model found

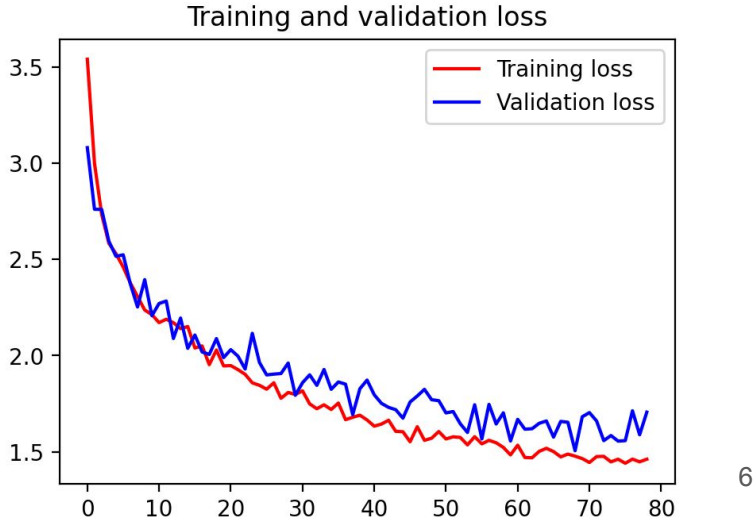
```
# define search space
convolution_layers_total_min = 2
convolution_layers_total_max = 6
convolution_layers_total_step = 1

filter_count_min = 32
filter_count_max = 64
filter_count_step = 32

kernel_size_min = 2
kernel_size_max = 4
kernel_size_step = 2

dense_layers_total_min = 9
dense_layers_total_max = 13
dense_layers_total_step = 1

units_count_min = 150
units_count_max = 250
units_count_step = 50
```

Baseline Model

```
model = tf.keras.
    # Flatten to
    tf.keras.laye
    tf.keras.laye

    tf.keras.laye
    tf.keras.laye
])

model.compile(los
              opt
              met
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_6 (Flatten)         (None, 22500)             0

 dropout_6 (Dropout)         (None, 22500)             0

 dense_12 (Dense)            (None, 128)               2880128

 dense_13 (Dense)            (None, 50)                6450

=================================================================
Total params: 2,886,578
Trainable params: 2,886,578
Non-trainable params: 0
_________________________________________________________________
```





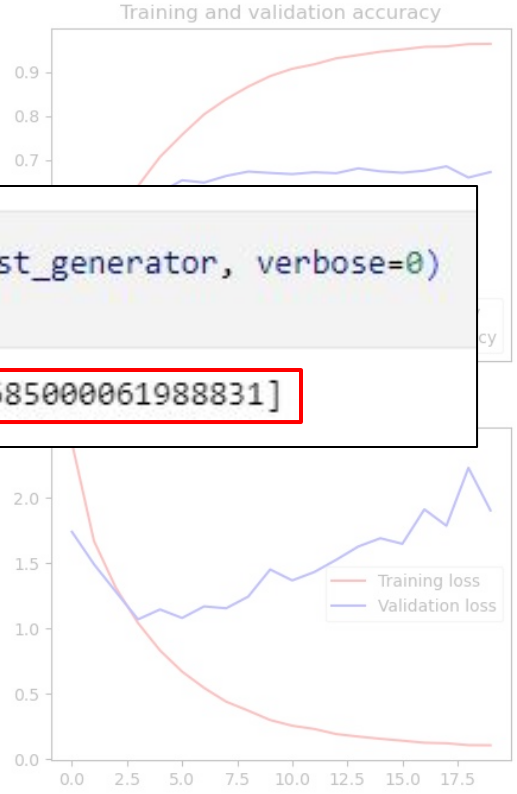
Results

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 149, 149, 64)	320
max_pooling2d (MaxPooling2D)	(None, 74, 74, 64)	0
conv2d_1 (Conv2D)		
conv2d_2 (Conv2D)		
max_pooling2d_1 (MaxPooling2D)		
flatten (Flatten)		
dense (Dense)	(None, 80)	6635600
dense_1 (Dense)	(None, 90)	7290
dropout (Dropout)	(None, 90)	0
dense_2 (Dense)	(None, 50)	4550

Total params: 6,676,552
Trainable params: 6,676,552
Non-trainable params: 0

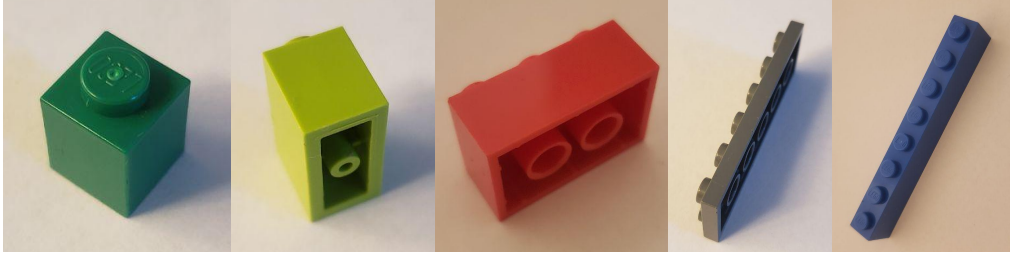
```
[79]  1  model.evaluate(test_generator, verbose=0)
      ✓ 9.3s

···   [1.984946846961975, 0.6685000061988831]
```



Problems

- Unexplainable Overfitting
  - ➔ Reason: Train- and Testgenerator contain different classes
  - ➔ Solution: shuffle data
- Can not load previously saved model (RuntimeWarning: Unexpected end-group tag: Not all data was converted)
  - ➔ solution: <3
- Defining a reasonable grid of Hyperparameters to prevent running out of memory
  - ➔ Solution: Trial-and-Error

Testing the model



```python
# prepare image to fit model input
image = np.array(image)
image = image.reshape((150, 150, 1))
image = image/255
image = np.expand_dims(image, axis=0)

# predict image
prediction = model.predict(image, verbose=0)
index_class_predicted = np.argmax(prediction)
keys=list(validation_generator.class_indices.keys())
class_predicted = keys[index_class_predicted]
class_predicted
```

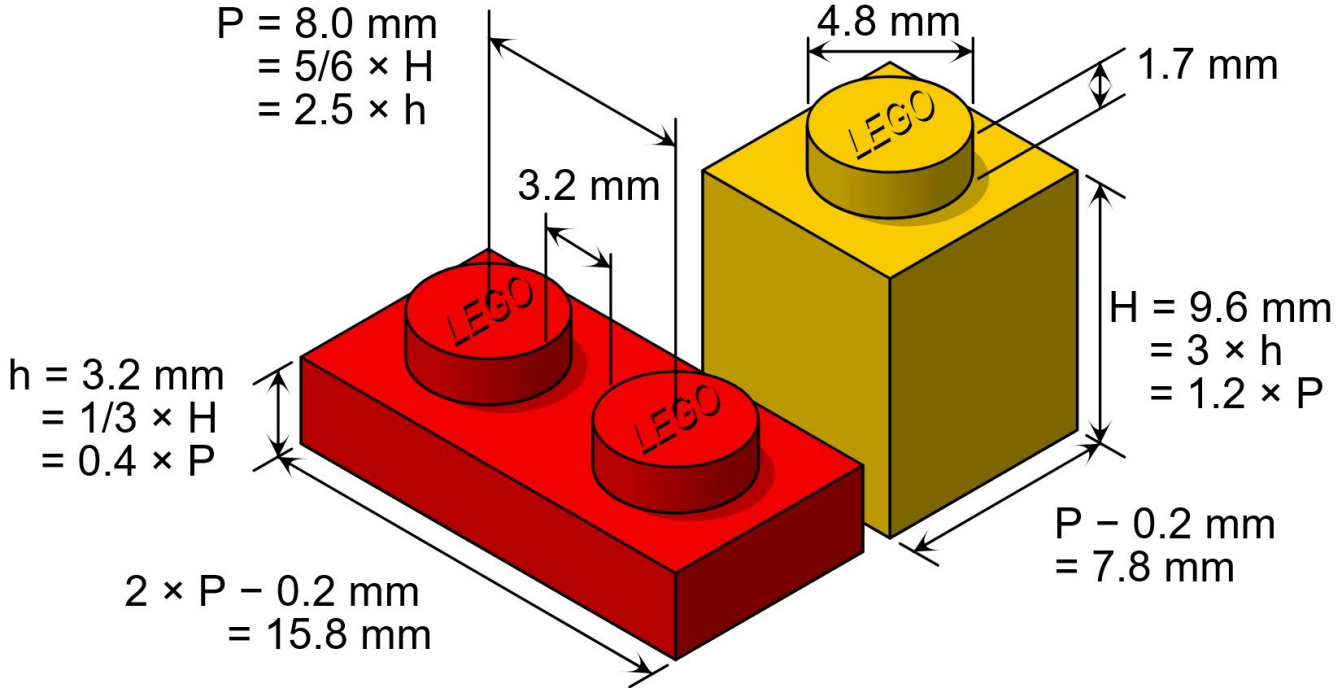
✓ 0.4s

```
'27925 flat tile round 2x2'
```

```python
# prepare image to fit model input
image = np.array(image)
image = image.reshape((150, 150, 1))
image = image/255
image = np.expand_dims(image, axis=0)

# predict image
prediction = model.predict(image, verbose=0)
index_class_predicted = np.argmax(prediction)
keys=list(validation_generator.class_indices.keys())
class_predicted = keys[index_class_predicted]
class_predicted
```

✓ 0.4s

```
'3001 brick 2x4'
```



https://upload.wikimedia.org/wikipedia/commons/1/1a/Lego_dimensions.svg

Thank you for your attention!