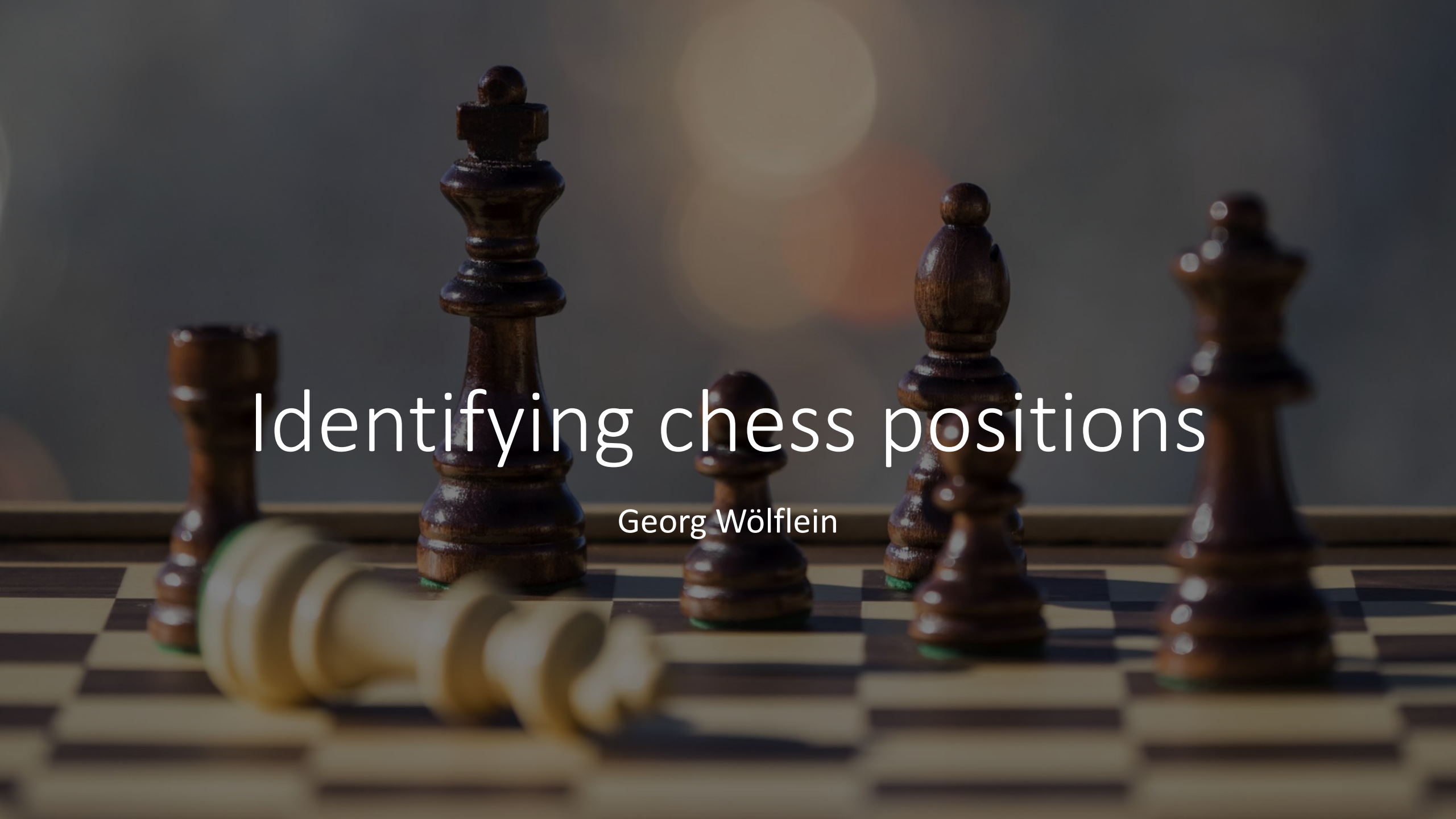


Identifying chess positions

Georg Wölflein

The goal

Give the computer an image of a chess board and it will recognize the position

The problem

13 classes

- Empty square
- 6 white pieces
- 6 black pieces

Inputs

- RGB image of the chess board
- Coordinates of the four corners

Output

- What piece is in each square

Approaches

Train an object detection model

Train an image classifier for the pieces. Knowing the four corner coordinates, extract each square and feed it into the image classifier

The dataset

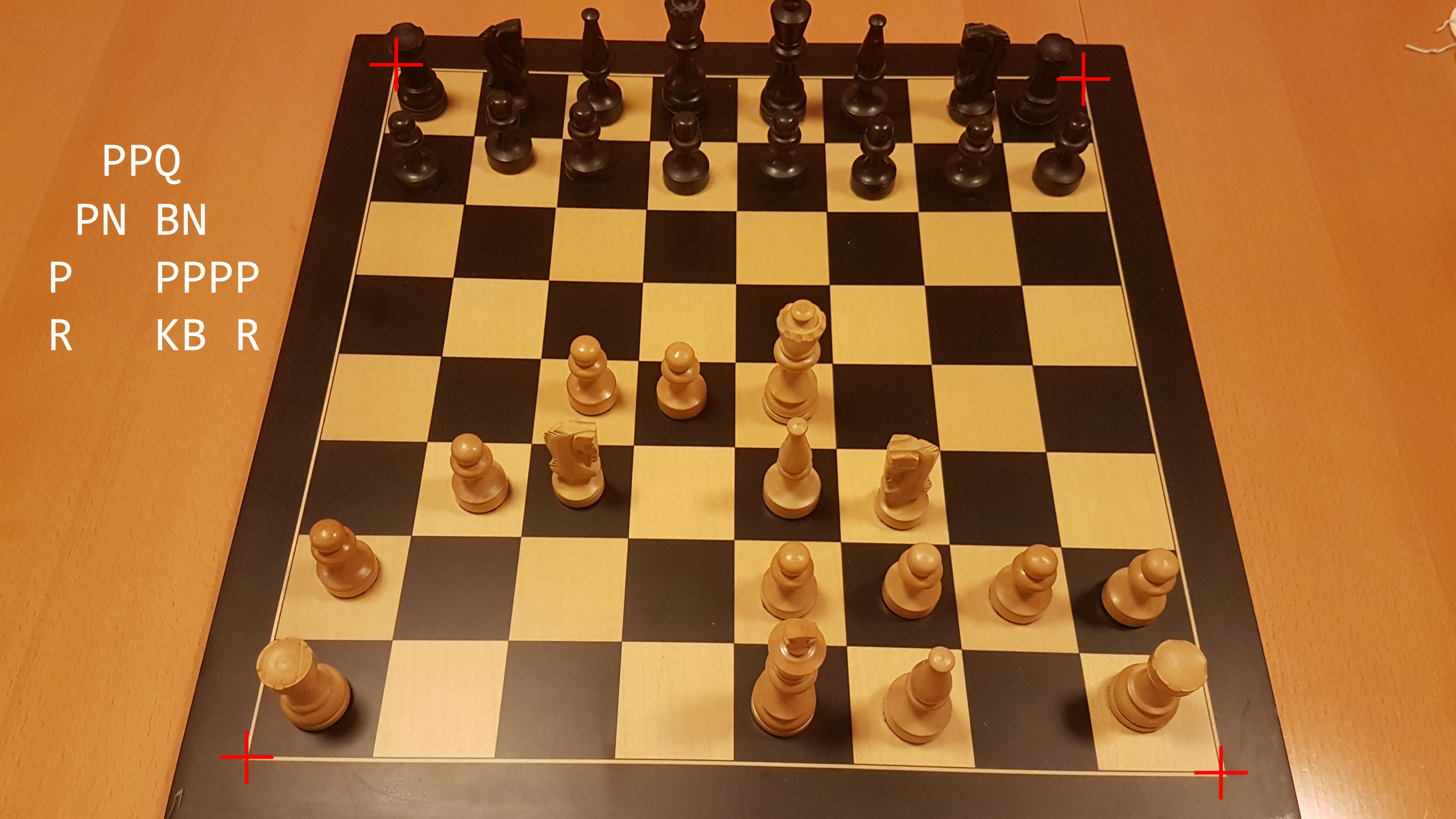
Data preparation

Manually labelled dataset

82 images of positions on a chess board

1 JSON file containing the (x, y) coordinates of all four corners for each image

82 text files with the positions for each image

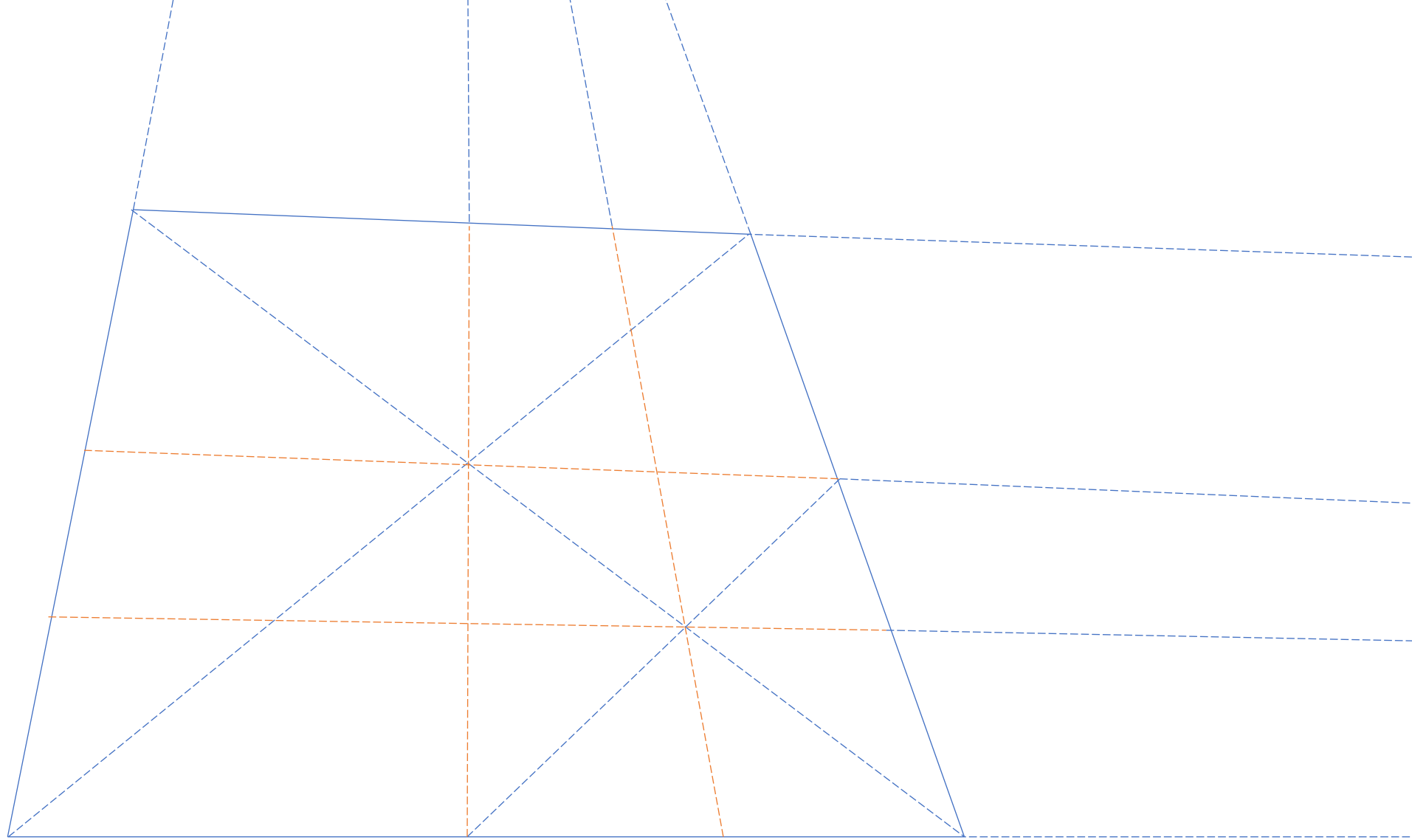

PPQ
PN BN
P PPPP
R KB R

Preparing the dataset

Extract the lower 32 squares into separate pictures

Use info from the text files to associate each picture with the correct class

Total of 82*32 = 2624 images



Unbalanced data

```
501 empty
125 white_pawn
125 black_pawn
 53 white_rook
 53 white_knight
 53 white_bishop
 53 black_rook
 53 black_knight
 53 black_bishop
 41 white_queen
 41 white_king
 41 black_queen
 41 black_king
```

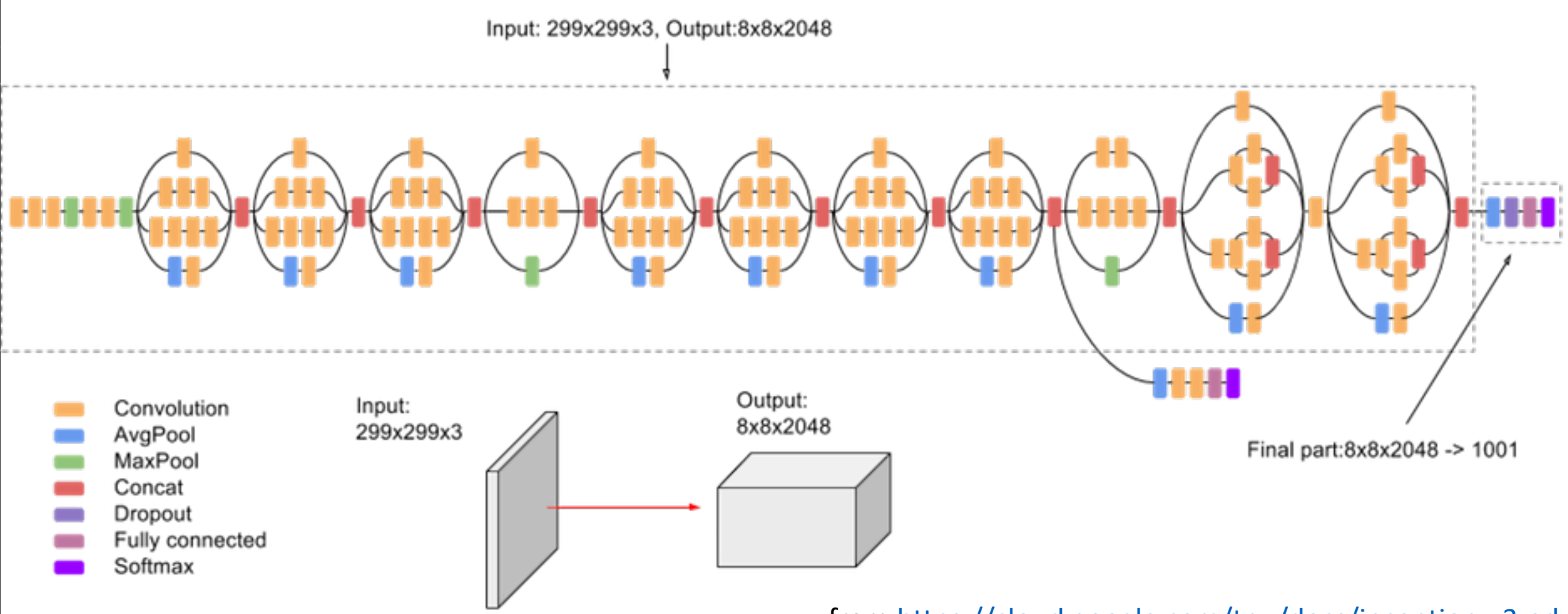
Dataset augmentation

Augment images (using imgaug) such that there are 500 per class

13*500 = 6500 images

Saved in a folder
(only needs to be done once)

```
seq = iaa.Sequential([
    iaa.Fliplr(0.5),
    iaa.PerspectiveTransform(scale=(0.01, 0.15)),
    iaa.LinearContrast((0.75, 1.5)),
    iaa.Multiply((0.8, 1.2)),
    iaa.Affine(
        scale={"x": (0.8, 1.2), "y": (0.8, 1.2)},
        translate_percent={"x": (-0.1, 0.1)},
        rotate=(-20, 20),
        shear=(-6, 6)
    )
])
```



from https://cloud.google.com/tpu/docs/inception-v3-advanced

The model

Inception-V3 with custom top

```
model = tf.keras.Sequential([
    inception_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dropout(.5),
    tf.keras.layers.Dense(64, activation="relu"),
    tf.keras.layers.Dropout(.2),
    tf.keras.layers.Dense(32, activation="relu"),
    tf.keras.layers.Dense(len(LABELS), activation="softmax")
])
```

The model

Inception-V3 with custom top

Training

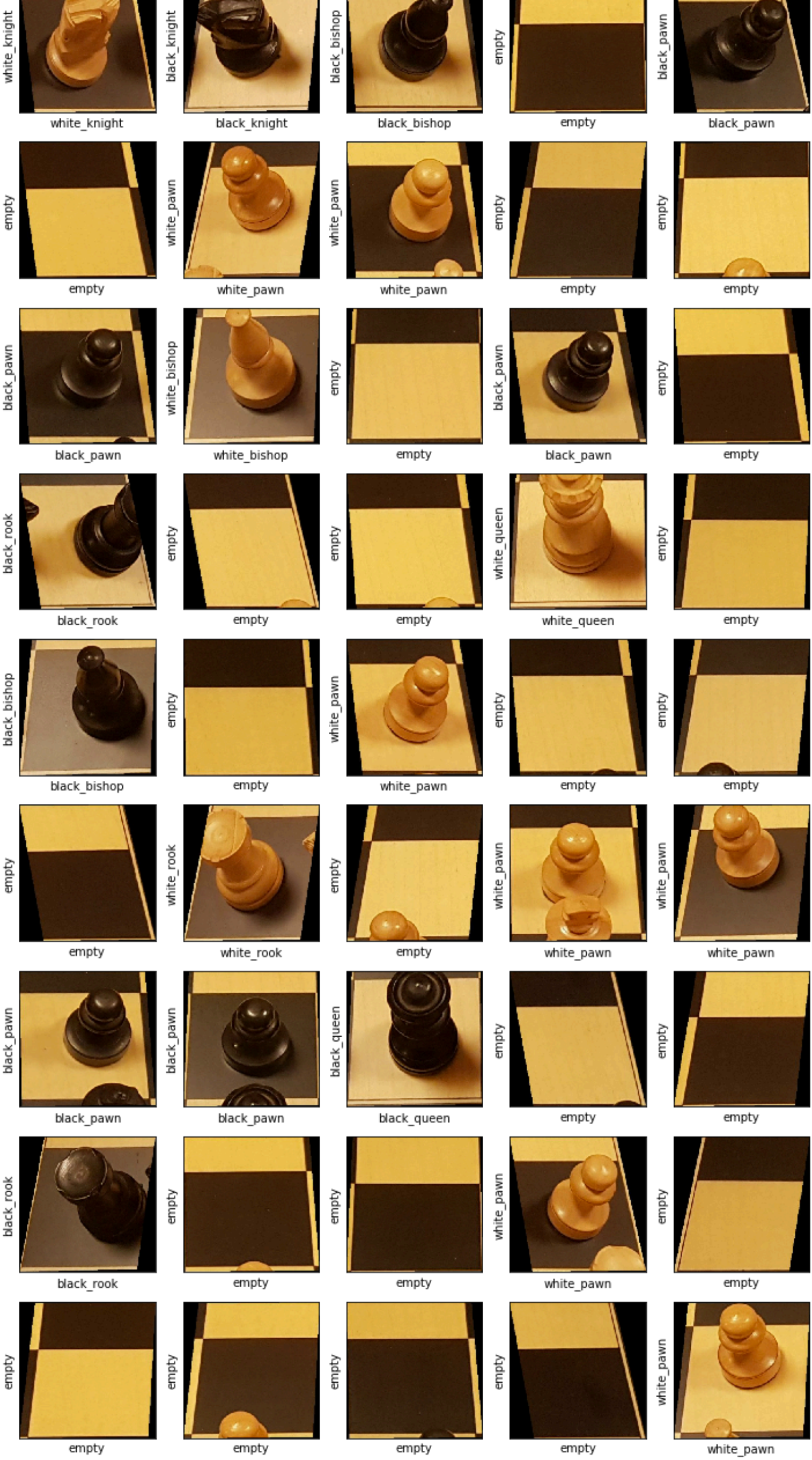
Batch size: 64

Steps per epoch: 6500 // 64 = 101

Optimizer: Adam

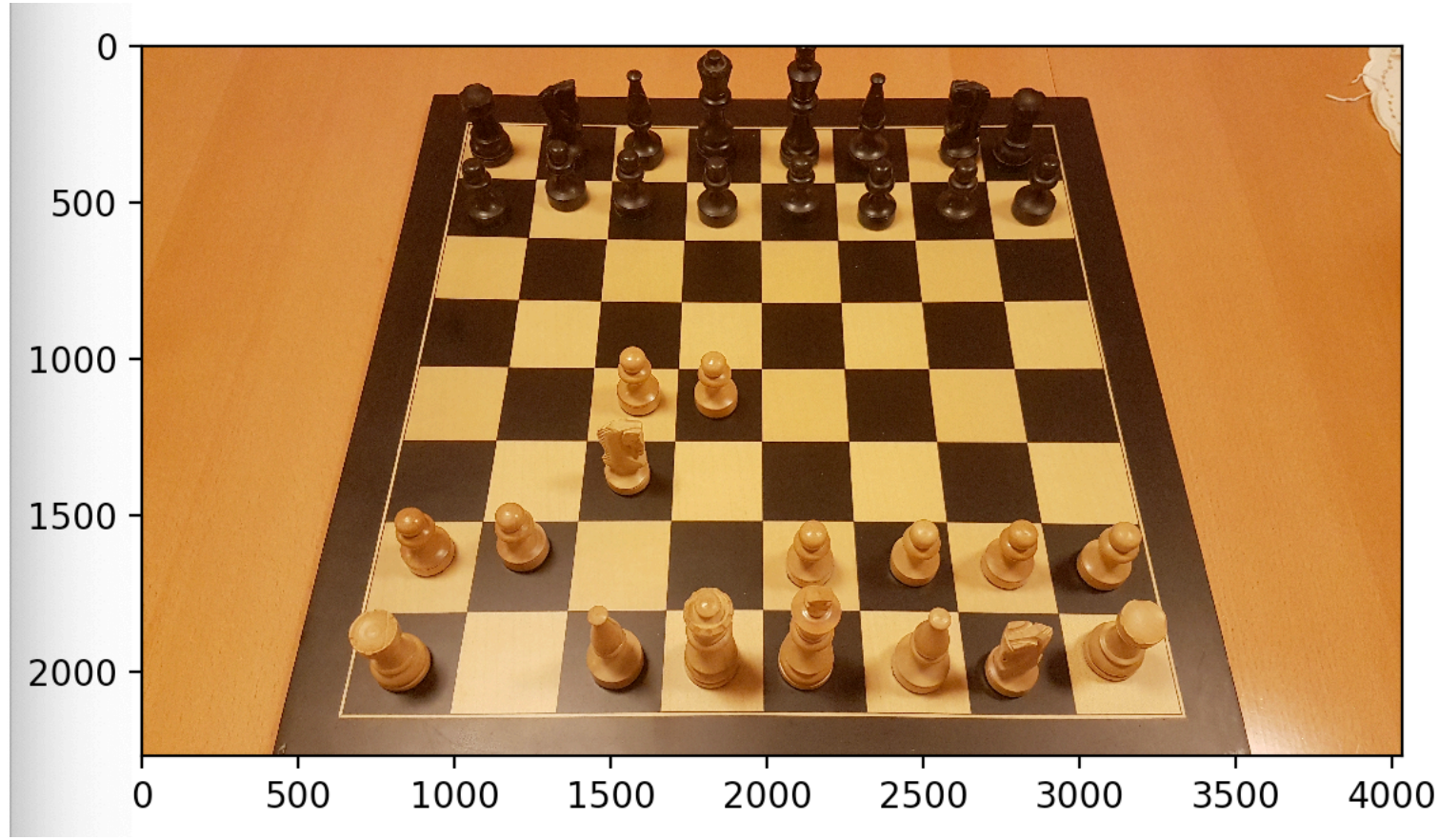
Loss: categorical cross-entropy

- Trained for 5 epochs @ .001 learning rate (only top layers)
  - Achieved 80% train and 70% validation accuracy
- Trained for 2 epochs @ .0001 learning rate (all layers)
  - Achieved >99% train and validation accuracy


white_knight
black_knight
black_bishop
empty
black_pawn
empty
white_pawn
white_pawn
empty
empty
black_pawn
white_bishop
empty
black_pawn
empty
black_rook
empty
empty
white_queen
empty
black_bishop
empty
white_pawn
empty
empty
empty
white_rook
empty
white_pawn
white_pawn
black_pawn
black_pawn
black_queen
empty
empty
black_rook
empty
empty
white_pawn
empty
empty
empty
empty
empty
white_pawn

Demonstration (on the test set)

```
File: data/images/test/w01.jpg
H | r n b k q b n b
G | p p p p q p p p
F | p     p   p p p
E |
D |     P P
C |     N
B | P P     P P P P
A | R   B Q K B N R
  +----------------
    1 2 3 4 5 6 7 8
Press enter to continue or 'q' to quit:
```



Known issues

- Towards the back of the board, empty squares are often classified as the piece that is below them
  - Squares at the back of the board are cropped with a greater height due to the perspective

Next steps

- Train an object detection model instead
  - Look at the coordinates of the bottom of the bounding boxes
- Train an additional binary classifier to detect empty squares
  - Taking as input only the tightly cropped square
  - Run the trained model only on the non-empty squares (*cropped as before*)

- Figure out a way to automatically detect the coordinates of the four corner points

Links

- GitHub repository: https://github.com/georgw777/chess
- Dataset: https://drive.google.com/file/d/1P7xMf4kQaAmuuLNYXKqVjpx_244zqWIU/view?usp=sharing (ZIP archive containing raw images, annotations, and augmented dataset)