Recognizing Chess pieces with a CNN

By Trent Conley

First attempt

Problems: dark board, dark table, computer did not recognize the difference between position of squares. Got a 7% accuracy for just the pawn location recognition

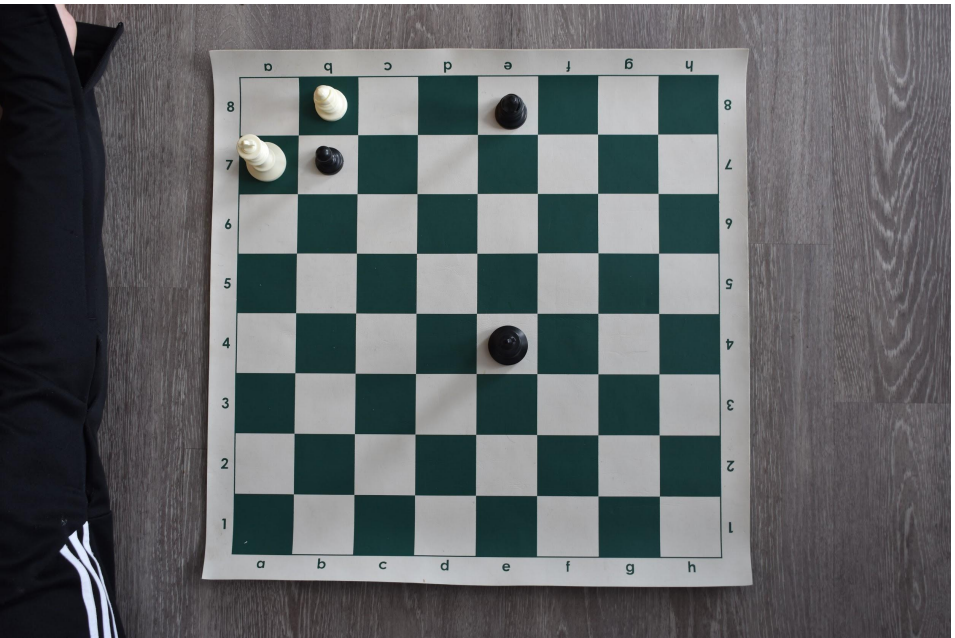
Lower angles mean more overlap of pieces

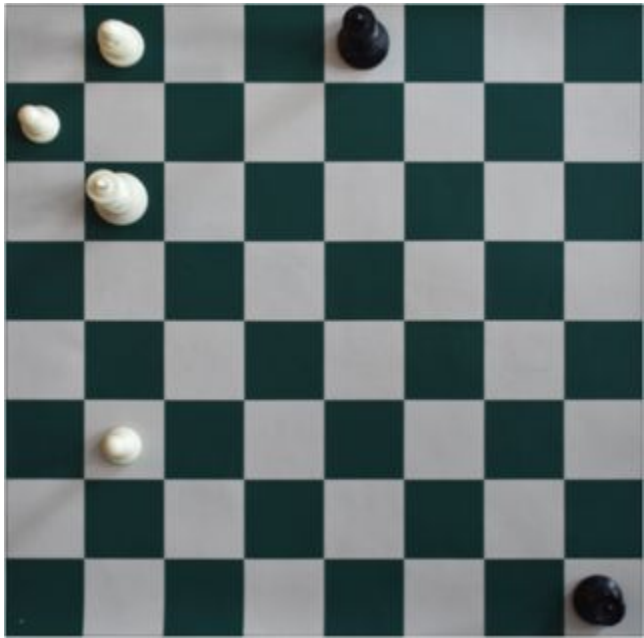
Used a CNN

Second attempt

Used dataset already built from https://github.com/samryan18/chess-dataset.git

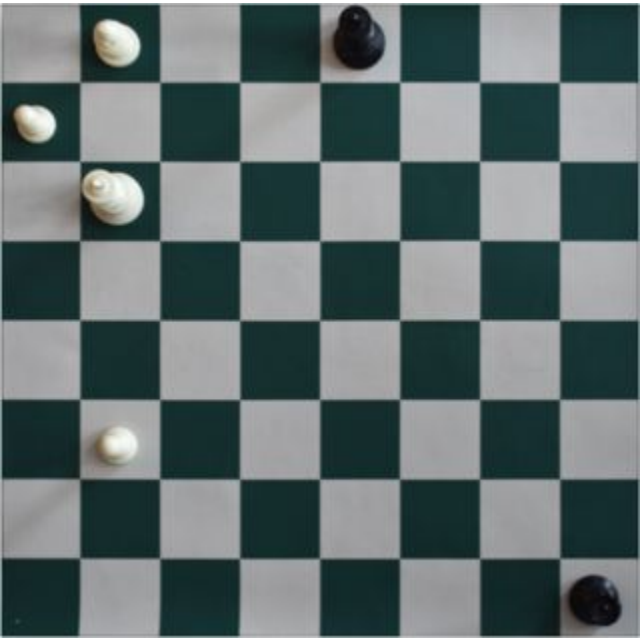
Example of a processed image on the right, not the same image





Breaking up the images

Wrote custom method to split processed image into individual pieces, and to worsen resolution so that train time was reduced from a couple days to around 5 hours.



Ended up with over
32,000 40x40 images

Results

Accuracy after 50 epochs: 0.9711019396781921

Relatively good, given that I had to manually prune images

Deleted 400 images that were classified wrong

Deleting bad images increased accuracy

Saving model

Did not want to train every time

Saved model onto json and h5 file

Json:

{"class_name": "Sequential", "config": {"name": "sequential_1", "layers": [{"class_name": "Conv2D", "config": {"name": "conv2d_1", "trainable": true, "batch_input_shape": [null, 40, 40, 4], "dtype": "float32", "filters": 48, "kernel_size": [3, 3], "strides": [1, 1], "padding": "same", "data_format": "channels_last", "dilation_rate": [1, 1], "activation": "linear", "use_bias": true, "kernel_initializer": {"class_name": "VarianceScaling", "config": {"scale": 1.0, "mode": "fan_avg", "distribution": "uniform", "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Activation", "config": {"name": "activation_1", "trainable": true, "dtype": "float32", "activation": "relu"}}, {"class_name": "Conv2D", "config": {"name": "conv2d_2", "trainable": true, "dtype": "float32", "filters": 48, "kernel_size": [3, 3], "strides": [1, 1], "padding": "valid", "data_format": "channels_last", "dilation_rate": [1, 1], "activation": "linear", "use_bias": true, "kernel_initializer": {"class_name": "VarianceScaling", "config": {"scale": 1.0, "mode": "fan_avg", "distribution": "uniform", "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Activation", "config": {"name": "activation_2", "trainable": true, "dtype": "float32", "activation": "relu"}}, {"class_name": "MaxPooling2D", "config": {"name": "max_pooling2d_1", "trainable": true, "dtype": "float32", "pool_size": [2, 2], "padding": "valid", "strides": [2, 2], "data_format": "channels_last"}}, {"class_name": "Dropout", "config": {"name": "dropout_1", "trainable": true, "dtype": "float32", "rate": 0.25, "noise_shape": null, "seed": null}}, {"class_name": "Flatten", "config": {"name": "flatten_1", "trainable": true, "dtype": "float32", "data_format": "channels_last"}}, {"class_name": "Dense", "config": {"name": "dense_1", "trainable": true, "dtype": "float32", "units": 10, "activation": "linear", "use_bias": true, "kernel_initializer": {"class_name": "VarianceScaling", "config": {"scale": 1.0, "mode": "fan_avg", "distribution": "uniform", "seed": null}}, "bias_initializer": {"class_name": "Zeros", "config": {}}, "kernel_regularizer": null, "bias_regularizer": null, "activity_regularizer": null, "kernel_constraint": null, "bias_constraint": null}}, {"class_name": "Activation", "config": {"name": "activation_3", "trainable": true, "dtype": "float32", "activation": "softmax"}}]}, "keras_version": "2.3.1", "backend": "tensorflow"}

H5 file too large, shows all of the saved weights, which are a lot

Testing for processed image taken from camera

Test board:

Terminal output: test accuracy = 1.0

```
Loaded model from disk
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 40, 40, 48)        1776
_________________________________________________________________
activation_1 (Activation)    (None, 40, 40, 48)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 38, 38, 48)        20784
_________________________________________________________________
activation_2 (Activation)    (None, 38, 38, 48)        0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 19, 19, 48)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 19, 19, 48)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 17328)             0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                173290
_________________________________________________________________
activation_3 (Activation)    (None, 10)                0
=================================================================
Total params: 195,850
Trainable params: 195,850
Non-trainable params: 0
_________________________________________________________________

Test accuracy: 1.0
Trents-MacBook-Pro:Secondus trentconley$
```

Future implications

App that allows you to predict the best move

Train CNN to recognized pieces taken from side

Opens up door to real-time image recognition like self-driving cars

Integration of technology with reality