Hi, I’m Kevin Fu, and this is my project on Automating Chess Move Notation.

[slide]

If you’ve ever watched a televised chess match, you may have noticed that the players in the game usually have something next to their chessboards:

[slide]

clipboards, with pens or pencils on top. They need clipboards to write down the moves of the game, so they can study the match later and verify that both players agree on the same move order.

However, those watching the broadcast follow along not on the actual board, but on a digital board to the side of the match, which a human expert must update as the match happens. If the broadcast doesn’t have an expert,

[slide]

the viewers themselves must figure out what’s happening on the board from the video, which can be challenging for inexperienced chess players. (Like me.)

Wouldn’t it be better if the process of move transcription was automated? Chess players would have an easier time studying their own games, and chess viewers could follow games without an expert guiding them. (pause)

[slide]

Well, thanks to Hough transforms, we can detect the straight lines of a chess board. And with convolutional neural networks, or CNNs, we can identify complex shapes from images, like chess pieces.

[slide]

By combining board detection from a Hough transform with piece recognition from a CNN, we’re able to transcribe chess moves from a video feed in real time.

[skip board detection slides]

If we gloss over the inner workings of the CNN at its heart, piece detection would be fairly simple: segment the board into squares, and take the CNN’s predictions for each square. However, we’ve added some overhead to reduce the number of squares the CNN has to check.

[slide]

By applying a homography transform to the four inner corners of the board, we can extract two valuable pieces of information: an orthophoto, and the estimated piece height.

The orthophoto is a top-down projection, which we then run Canny edge detection on to get an outline of the pieces. The program then counts the number of edge pixels in the upper region of each square (as shown in green). If the number of edge pixels passes a certain threshold, the square is flagged for review by the CNN. If not, it’s declared empty, no CNN required.

The estimated piece height, given using a technique called pose estimation, gives us the segmented squares for both training and testing input into the CNN.

[slide]

The red box drawn shows how the program takes double the height of the vertical green line shown as the maximum possible height for any piece, then segments the image accordingly (as shown in the middle). For training purposes, we run the image through robust data augmentation to increase the CNN’s resistance to different lighting and angle conditions.

[slide]

Finally, we transfer learn off a modified CNN structure called ResNet, which has the industry-leading accuracy on the ImageNet competition, a broad image classification problem. By starting learning using the weights ResNet learned for ImageNet, we are fine-tuning the model’s image classification to specifically classify chess pieces.

As shown by the arrow and the confusion matrix, the CNN is highly effective. The y-axis of the confusion matrix is “true label,” and the x-axis is the CNN’s “predicted label.” A perfect CNN would have only blue down the diagonal, and while there are a few errors, this CNN is nearly flawless.

(On the validation dataset it’s been given at least; there are some problems with that.)

[slide]

\*show live demo\*