# Machine Learning: AI Learns To Play Tetris with Convolutional Neural Network

Trong giới thiệu này, tôi sẽ trình bày khái niệm tạo ra một AI Bot chơi Tetris như một con người thực sự. Nó không hoàn hảo 100%, nhưng khá tốt.
Tôi đã sử dụng Machine Learning với Convolutional Neural Network và python cho việc lập trình.

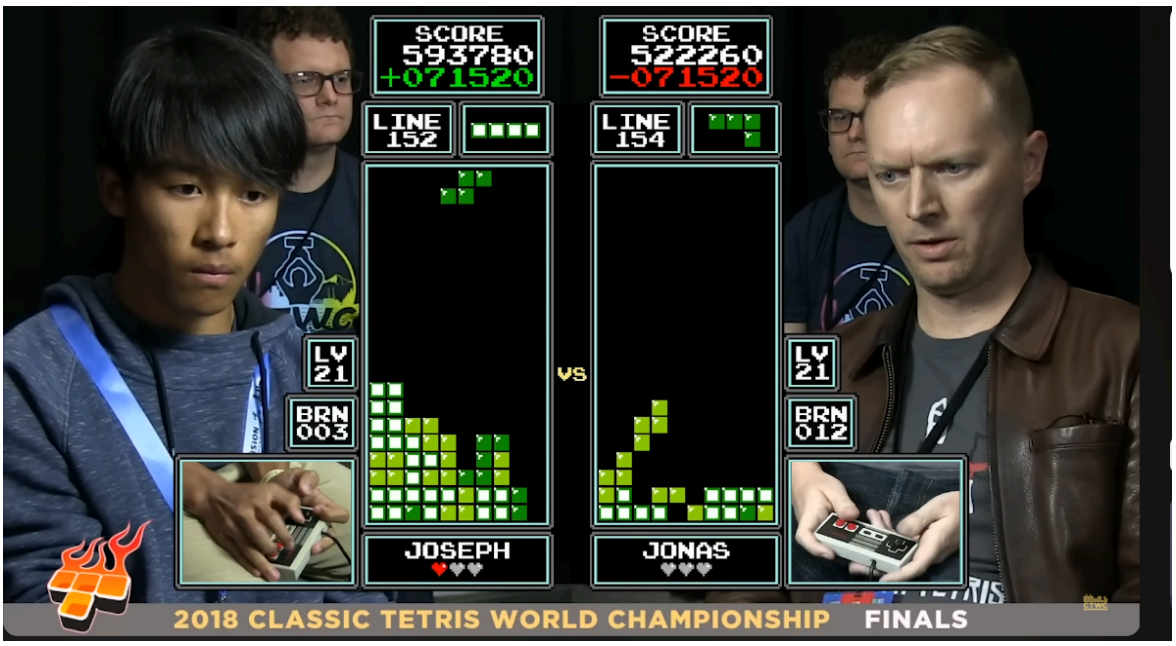
## 1. Thu Thập Data

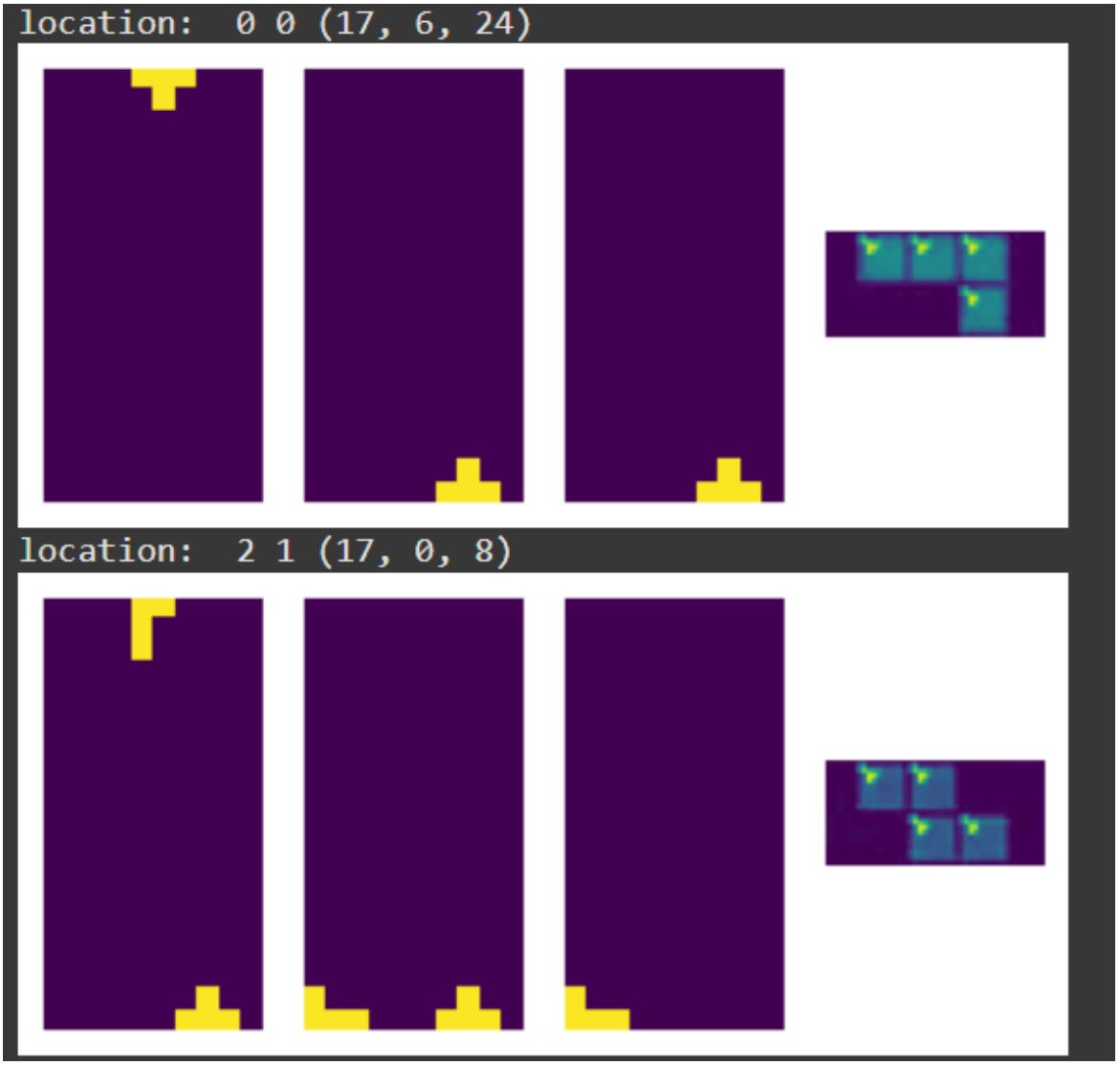
Để đào tạo mạng, tôi cần một tập dữ liệu chất lượng cao về các cấu hình bảng khác nhau được mô tả bằng hình ảnh có nhãn tương ứng, trong đó:

- Hình ảnh là ảnh chụp nhanh của bảng.
- Nhãn là hành động thể hiện vị trí cột cuối cùng và vòng quay của một khối đã chơi trên bảng đó.

Tôi lấy dữ liệu này bằng cách nào?

Vâng, có một kênh Youtube với rất nhiều video chiếu các trận đấu Giải vô địch thế giới Tetris. Ở đó, tất cả các đối thủ đều chơi Tetris ở cấp độ cao nhất và mắc một số lỗi nhỏ. Bên cạnh đó, họ đặt mục tiêu ghi được nhiều điểm nhất bằng cách xóa bốn hàng cùng một lúc mọi lúc. Vì vậy, các trận đấu này là nguồn dữ liệu Tetris chất lượng cao tuyệt vời! Bạn chỉ cần thu thập dữ liệu bằng cách nào đó. Và đó là ý tưởng của tôi về việc lấy dữ liệu đào tạo.




Dựa trên quá trình xử lý hình ảnh của từng khung hình video, tôi tiến hành phân tích sự khác biệt giữa màu pixel của khung hình trước và khung hình hiện tại

Đây là link colab tôi tiến hành xử lí video và thu bộ dữ liệu đánh nhãn:
[process_data.ipynb - Colab]

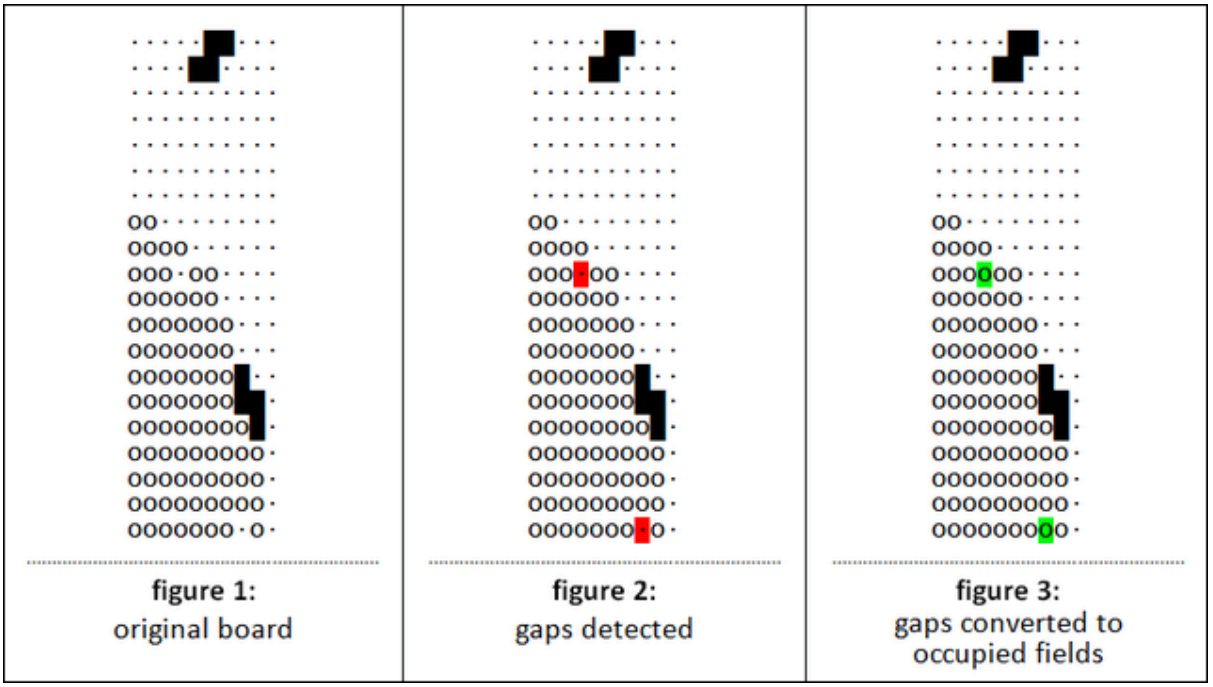
## 2. Augmenting Data

Trong quá trình thử nghiệm tôi nhận thấy bộ dữ liệu hiện tại không đủ để đào tạo mạng tôi ước lượng nó cần khoảng 1.000.000 bản ghi mới ổn.
Do đó trước khi build model và train tôi đã tiến hành tiền xử lí dữ liệu cũng như tăng cường dữ liệu giúp mô hình học tốt hơn các đặc trưng của ảnh đầu vào

Nhưng trước khi xử lý, tôi đã thay đổi tất cả các bảng chơi bằng cách chuyển đổi tất cả các khoảng trống thành các ô đã chiếm dụng.

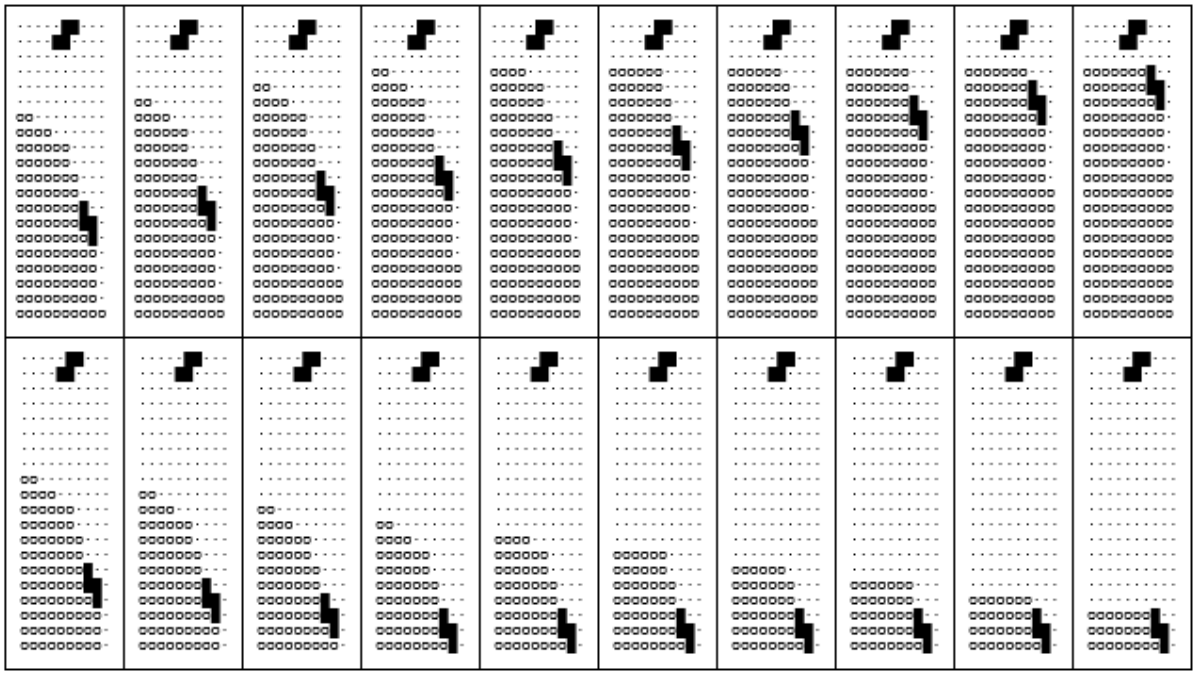
Tại sao hả?
Tại vì trong quá trình thử nghiệm nhiều lần tôi thấy việc thay thế như vậy giúp mạng CNN tập trung nhận diện vào vị trí cần đặt khối tốt hơn thay vì phải quá quan tâm vào những khoảng trống không quan trọng => từ đó giúp tăng độ chính xác lúc đặt khối tetrominos vào bảng

Hình ảnh bên dưới thể hiện điều đó:




Hình 1 cho thấy một bảng 20×10 gốc được lấy từ video.
Sau đó, trong hình 2, chúng ta thấy có hai khoảng trống trên bảng này.
Cuối cùng, trong hình 3, các khoảng trống được chuyển đổi thành các ô đã chiếm dụng.

Bây giờ, chúng ta hãy xem xét khái niệm tăng cường dữ liệu. Sử dụng bảng được hiển thị trong hình sau, ta đã tạo ra các bảng này:



10 bảng đầu tiên được tạo bằng cách chèn các dòng mới ở dưới cùng.
10 bảng dưới được tạo bằng cách xóa từng dòng một.

Ok, giờ tôi đã có 1 lượng data đủ lớn để đào tạo mô hình CNN!!!

## 3. Xây Dựng CNN Model

Đây là mô hình sequential:

Để chạy mô hình, tôi đã sử dụng các tham số sau:

Optimizer: Adam với tốc độ học là 0.001
Hàm mất mát: Categorical Cross entropy
Đo lường đánh giá: Accuracy

### 3.1. Input

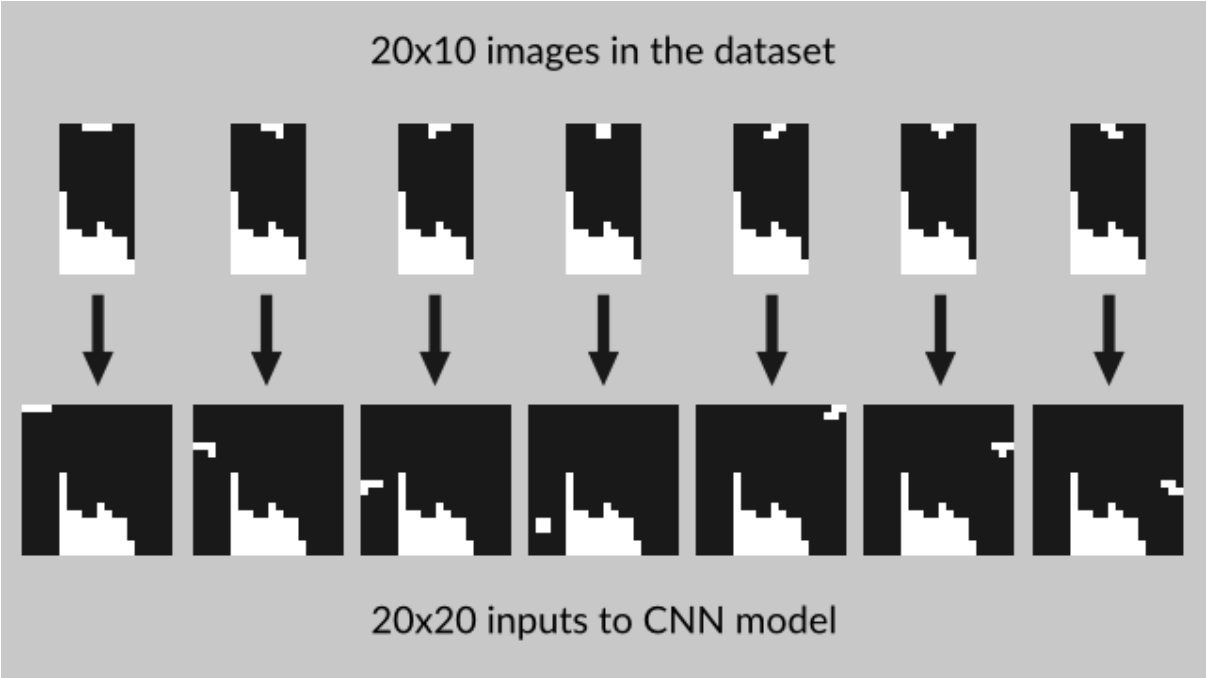
Ban đầu khi xử lí đầu vào tôi tiến hành cho trực tiếp ảnh 20x10 nhưng đã gặp vấn đề về hội tụ mặc dù đã thay đổi các siêu tham số.

Sau một thời gian nhìn nhận thì tôi nhận thấy việc sử dụng ảnh 20x10 và đặt các loại khối sử dụng trong bảng chơi ở cùng 1 vị trí gây khó khăn cho việc nhận diện khối sử dụng ở mỗi trạng thái game.

Do đó một ý tưởng đó là mở rộng bảng chơi qua trái vào phải mỗi bên 5 ô và thay đổi vị trí đặt mặc định của từng khối. Cụ thể hơn:

Thay vì giữ một khối tetromino ở vị trí ban đầu ở giữa hàng đầu tiên, tốt hơn là đặt từng khối vào vị trí cụ thể của chúng trong các cột bổ sung.

Có vẻ như CNN nhận dạng các quân cờ khác nhau tốt hơn khi được đặt ở các vị trí riêng biệt, hình ảnh dưới đây thể hiện ý tưởng đó



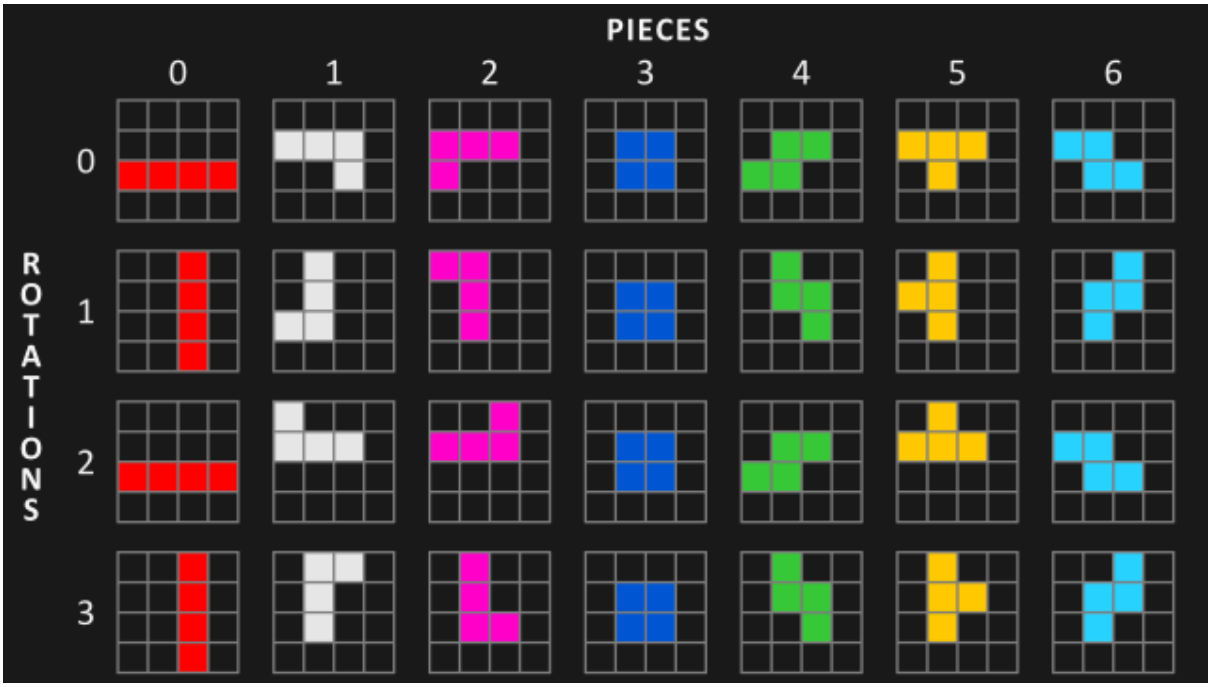

### 3.2. Output

Đầu ra từ mô hình là một trong 44 hành động có thể (nhãn).

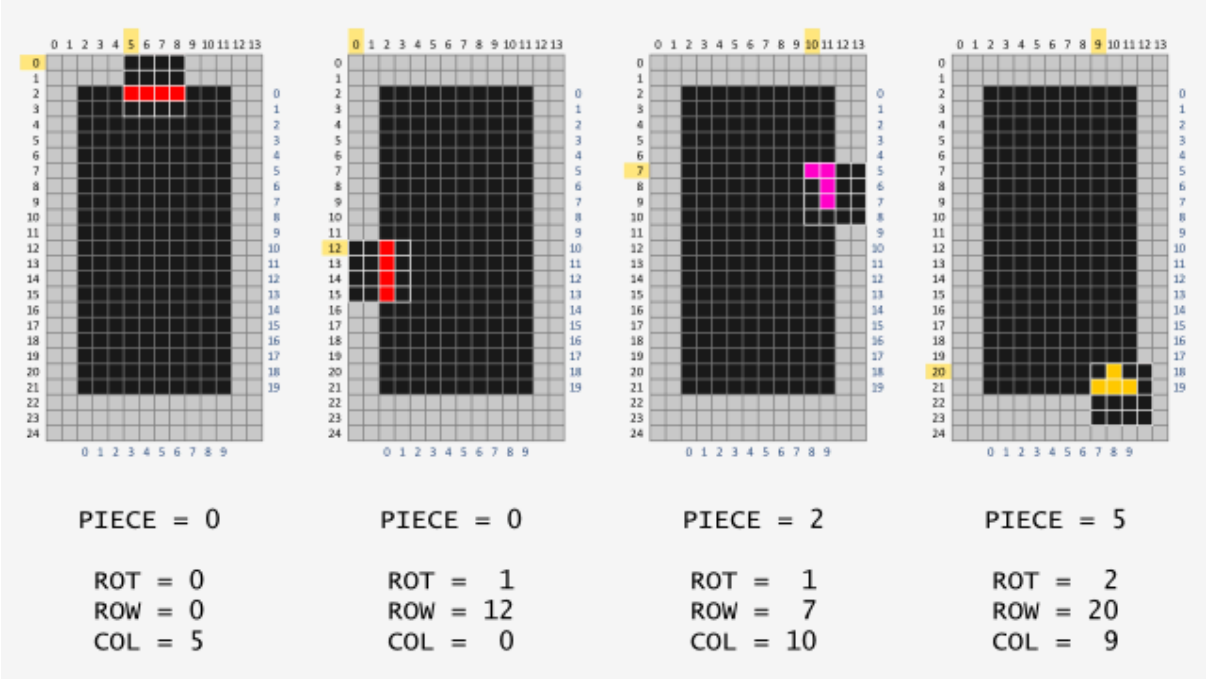
Và tại sao lại có 44 hành động đầu ra?

Khá đơn giản, mỗi hành động biểu thị sự kết hợp giữa vị trí cột cuối cùng và vòng quay cho khối tetromino được chơi trên cấu hình bàn cờ hiện tại.

Vậy trước tiên chúng ta hãy xem xét cấu trúc thân của từng khối tetromino. Chúng được xây dựng từ các khối 4×4. Bên cạnh đó, mỗi khối tetromino có 4 vòng quay, như thể hiện trong hình ảnh bên dưới:




Ok, giờ quay lại với bảng chơi thì ta cần đặt các quân cờ vào như nào mà có tận 11 vị trí đặt
Điều đó được thể hiện qua ảnh




để đặt bất kỳ khối tetromino nào vào vị trí cuối cùng của nó, chúng ta cần 11 cột (được đánh dấu từ 0 đến 10).

Vì mỗi khối tetromino có 4 vòng quay (được đánh dấu từ 0 đến 3), nên tổng cộng có 44 hành động (11 * 4).

## 4. Training CNN Model

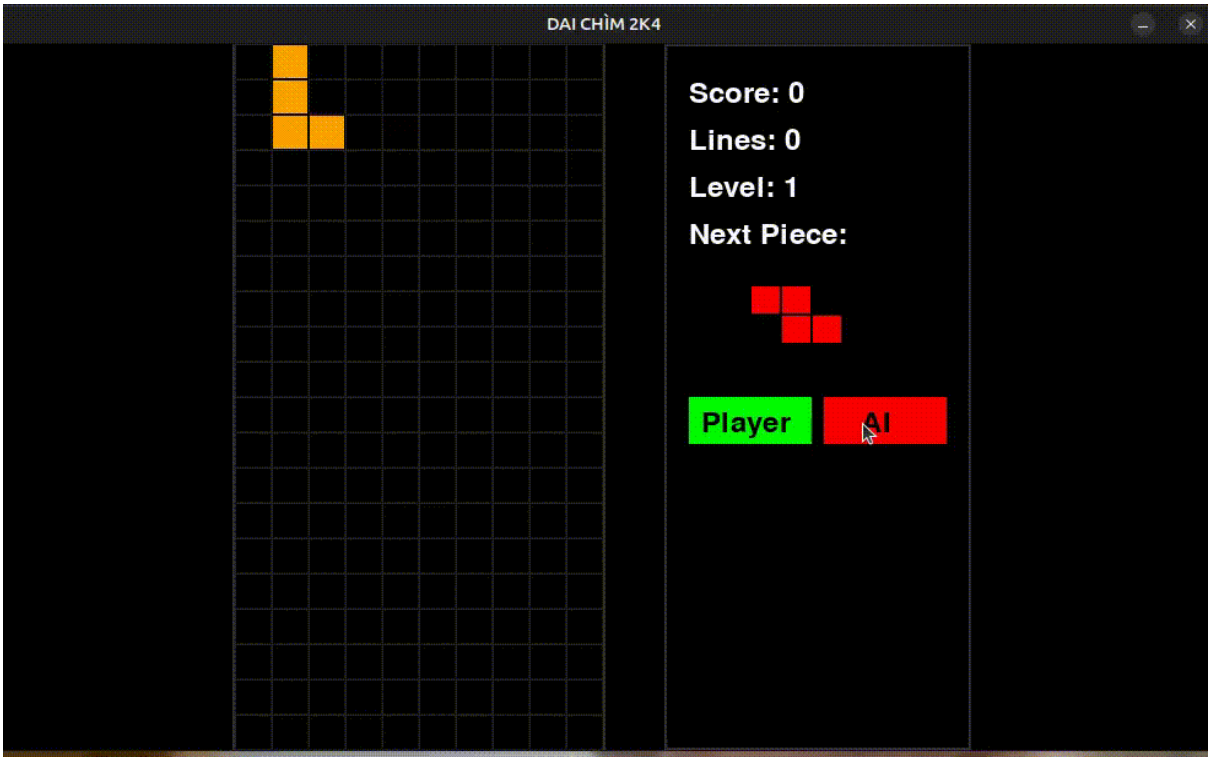
Sau khi xử lí dữ liệu ta tiến hành train model với 1% dữ liệu để test và 99% dùng để train

```
X_train, X_test, y_train, y_test = train_test_split(dataset, labels, test_size=0.01, random_state=42)

# Chuyển đổi dữ liệu thành tensor PyTorch
X_train = torch.tensor(X_train, dtype=torch.float32).unsqueeze(1)  # Thêm chiều kênh: (batch_size, 1, 20, 20)
X_test = torch.tensor(X_test, dtype=torch.float32).unsqueeze(1)

y_train = torch.tensor(y_train, dtype=torch.long)  # Nhãn phải là dtype long cho classification
y_test = torch.tensor(y_test, dtype=torch.long)

# Tạo DataLoader cho train và test
class TetrisDataset(Dataset):
    def __init__(self, data, labels):
        self.data = data
        self.labels = labels

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        return self.data[idx], self.labels[idx]

train_dataset = TetrisDataset(X_train, y_train)
test_dataset = TetrisDataset(X_test, y_test)
```

Tôi tiến hành đào tạo mạng khoảng 10000 lần sau đó lưu trọng số của mạng và cứ đào tạo 100 lần/ lần chạy điều này đem lại độ chính xác khoảng 97.34%

```
optimizer = optim.Adam(model.parameters(), lr=0.0005)  # Optimizer Adam
for epoch in range(10000):
    model.train()  # Đặt mạng ở chế độ huấn luyện
    running_loss = 0.0
    correct = 0
    total = 0
    for images, labels in train_loader:
        # Chuyển dữ liệu vào GPU
        images, labels = images.to(device), labels.to(device)

        optimizer.zero_grad()  # Làm mới gradient
        # Forward pass
        outputs = model(images)
        # print(outputs)
        # print(labels)
        # Tính loss
        loss = criterion(outputs, labels)

        # Backward pass và cập nhật weights
        loss.backward()
        optimizer.step()

        running_loss += loss.item()  # Cộng dồn loss
        _, predicted = torch.max(outputs, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()

    print(f'Epoch {epoch+1}/{epochs}, Loss: {running_loss/len(train_loader):.4f}, Accuracy: {100 * correct / total:.2f}%')
```

Cuối cùng kết quả cũng khá ấn tượng khi tôi sử dụng mô hình này cho game tôi xây dựng từ trước:




Tuy nhiên, nó không phải lúc nào cũng bất khả chiến bại và đôi khi thua cuộc rất nhanh, nhưng nhìn chung nó chơi cũng khá tốt

Tôi thấy điều này cũng dễ hiểu bởi nó học theo cách chơi của con người và việc sử dụng CNN trong môi trường mà việc đặt sai 1 khối có thể gây bất lợi cho những lần đặt khối sau cũng là điều dễ hiểu, nếu muốn nó chơi thực sự tốt tôi khuyến khích bạn nên tìm đọc về Deep Q learning - một thuật toán học tăng cường khá nổi tiếng.

Nếu bạn muốn thử nghiệm thì đây là link tôi thực hiện việc tiền xử lí dữ liệu và train mô hình:
CNN_TETRIS

## 5. Tổng Kết

Mục tiêu của dự án này là tạo ra một trí tuệ nhân tạo (AI) học cách chơi Tetris bằng mạng nơ-ron tích chập (CNN).

Đây không phải là ý tưởng dễ dàng do thách thức lớn nhất tôi đối mặt là việc làm sao tạo ra 1 tập dữ liệu đào tạo với chất lượng cao chưa kể việc dạy cho mạng hiểu cách chơi cần một lượng rất lớn dữ liệu đào tạo

Vâng, tôi đã dành khoảng 90% cho việc xử lí dữ liệu từ video trên youtube cũng như việc tiền xử lí dữ liệu còn việc xây dựng model CNN cũng như code ra game tetris tôi nghĩ không quá khó để thực hiện

Tuy nhiên không có gì đảm bảo một mô hình học có giám sát có thể đảm bảo độ chính xác lên đến 100% được và mô hình tôi trình bày ở đây cũng vậy nhưng có thể nói nó chơi khá tốt vượt ngoài kì vọng của tôi(mặc dù có những tình huống để thua rất buồn cười)

Tóm lại, với một tập dữ liệu lớn và chất lượng cao, chúng ta có thể dạy mạng nơ-ron tích chập chơi Tetris khá tốt tuy nhiên khá tốn công sức cho việc xử lí dữ liệu ảnh do đó nếu bạn đọc có hứng thú có thể tham khảo về Deep Q learning(RL).

Nếu thấy hay cho tôi xin 1 star, cảm ơn!!!!!!!!!!