```
!pip install python-chess torchvision
import chess
import chess.pgn
import chess.engine
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from google.colab import files
from torchvision import models
import random
import time
```

Download and set up Stockfish

```
!apt-get install -y stockfish
ENGINE_PATH = "/usr/games/stockfish"
engine = chess.engine.SimpleEngine.popen_uci(ENGINE_PATH)
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
stockfish is already the newest version (14.1-1).
0 upgraded, 0 newly installed, 0 to remove and 29 not upgraded.
```

Defining DL Model (Feed-Forward Neural Network)

```
class ChessEvaluator(nn.Module):
    def __init__(self):
        super(ChessEvaluator, self).__init__()
        self.fc1 = nn.Linear(8, 16)
        self.fc2 = nn.Linear(16, 5)

    def forward(self, x):
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x

# Model instance
model = ChessEvaluator()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Function to analyze chess game

```
def analyze_game_fast(pgn_file):
    pgn = open(pgn_file)
    game = chess.pgn.read_game(pgn)
    board = game.board()
    move_data = []

    start_time = time.time()

    for move in game.mainline_moves():
        board.push(move)
        analysis = engine.analyse(board, chess.engine.Limit(depth=6))
        centipawn_loss = analysis["score"].relative.score()
        move_data.append((board.fen(), centipawn_loss))

    print(f"Analysis completed in {time.time() - start_time:.2f}
seconds")
    return move_data
```

Function to upload file

```
def upload_pgn():
    uploaded = files.upload()
    filename = list(uploaded.keys())[0]

    # Check if the uploaded file is a PGN file
    if not filename.lower().endswith('.pgn'):
        raise ValueError("Invalid file format! Please upload a PGN
file.")

    return filename
```

Function to generate feedback

```
def generate_feedback(move_data):
    feedback = []
    thresholds = [0, 50, 100, 200]

    for _, centipawn_loss in move_data:
        if centipawn_loss is None:
            feedback.append("Move not analyzed.")
            continue

        if centipawn_loss == 0:
            feedback.append("Excellent move!")
        elif centipawn_loss < thresholds[1]:
            feedback.append("Good move but could be better.")
        elif centipawn_loss < thresholds[2]:
            feedback.append("Inaccuracy detected.")
        elif centipawn_loss < thresholds[3]:
            feedback.append("Mistake! Review this move carefully.")
        else:
            feedback.append("Blunder! Significant mistake made.")

    return feedback
```

Monte Carlo Tree Search (MCTS) for move selection

```
def mcts_move_selection(board, simulations=10):
    legal_moves = list(board.legal_moves)
    if not legal_moves:
        return None

    move_scores = {}

    for move in legal_moves:
        board.push(move)
        try:
            # Evaluate move using Stockfish at low depth
            analysis = engine.analyse(board,
chess.engine.Limit(depth=4))
            score = analysis["score"].relative.score() or 0
            move_scores[move] = score
        except:
            move_scores[move] = -1000  # Penalize errors
        board.pop()

    # □ Sort moves by highest score & return the best
    best_move = max(move_scores, key=move_scores.get)
    return best_move
```

Defining run analysis

```
def run_analysis():
    print("Upload your PGN file:")
    pgn_file = upload_pgn()
    move_data = analyze_game_fast(pgn_file)
    feedback = generate_feedback(move_data)
    for i, comment in enumerate(feedback):
        print(f"Move {i+1}: {comment}")

import os
os.listdir()

['.config', 'sample_data']
```

Run analysis

```
run_analysis()

Upload your PGN file:

<IPython.core.display.HTML object>

Saving Malakhov.pgn to Malakhov.pgn
Analysis completed in 0.32 seconds
Move 1: Good move but could be better.
Move 2: Good move but could be better.
Move 3: Good move but could be better.
Move 4: Good move but could be better.
Move 5: Good move but could be better.
Move 6: Good move but could be better.
Move 7: Good move but could be better.
Move 8: Good move but could be better.
Move 9: Good move but could be better.
Move 10: Good move but could be better.
Move 11: Good move but could be better.
Move 12: Inaccuracy detected.
Move 13: Good move but could be better.
Move 14: Inaccuracy detected.
Move 15: Good move but could be better.
Move 16: Inaccuracy detected.
Move 17: Blunder! Significant mistake made.
Move 18: Inaccuracy detected.
Move 19: Good move but could be better.
Move 20: Good move but could be better.
Move 21: Good move but could be better.
Move 22: Good move but could be better.
Move 23: Good move but could be better.
Move 24: Good move but could be better.
Move 25: Good move but could be better.
Move 26: Blunder! Significant mistake made.
Move 27: Good move but could be better.
Move 28: Inaccuracy detected.
Move 29: Good move but could be better.
Move 30: Good move but could be better.
Move 31: Good move but could be better.
Move 32: Inaccuracy detected.
Move 33: Good move but could be better.
Move 34: Good move but could be better.
Move 35: Mistake! Review this move carefully.
Move 36: Good move but could be better.
Move 37: Inaccuracy detected.
Move 38: Good move but could be better.
Move 39: Inaccuracy detected.
Move 40: Inaccuracy detected.
Move 41: Good move but could be better.
Move 42: Mistake! Review this move carefully.
Move 43: Good move but could be better.
Move 44: Inaccuracy detected.
Move 45: Good move but could be better.
Move 46: Mistake! Review this move carefully.
Move 47: Good move but could be better.
Move 48: Mistake! Review this move carefully.
Move 49: Good move but could be better.
Move 50: Blunder! Significant mistake made.
Move 51: Good move but could be better.
Move 52: Blunder! Significant mistake made.
Move 53: Good move but could be better.
Move 54: Blunder! Significant mistake made.
Move 55: Good move but could be better.
Move 56: Blunder! Significant mistake made.
Move 57: Good move but could be better.
```