In [ ]:
```
import pandas as pd

# Load the dataset
file_path = '/content/Student_performance_data.csv'
student_gpa_data = pd.read_csv(file_path)

# Display the first few rows of the dataset to understand its structure
student_gpa_data.head()
```

Out[ ]:

	StudentID	Age	Gender	Ethnicity	ParentalEducation	StudyTimeWeekly	Absences	Tut
0	1001	17	1	0	2	19.833723	7	
1	1002	18	0	0	1	15.408756	0	
2	1003	15	0	2	3	4.210570	26	
3	1004	17	1	0	3	10.028829	14	
4	1005	17	1	0	2	4.672495	17	

In [ ]:
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from sklearn.metrics import mean_squared_error
import numpy as np

# Prepare the data
X = student_gpa_data.drop(columns=['StudentID', 'GPA', 'GradeClass'])
y = student_gpa_data['GPA']

# Normalize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, ran

# Store results
results = []

# Experiment 1: A single Dense Hidden Layer
model1 = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(1)
])
model1.compile(optimizer='adam', loss='mse')
model1.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0, validation_data=(
pred1 = model1.predict(X_test, verbose=0)
mse1 = mean_squared_error(y_test, pred1)
results.append(['Single Dense Layer', mse1])
```

```
# Experiment 2: Three Dense Hidden Layers
model2 = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(64, activation='relu'),
    Dense(1)
])
model2.compile(optimizer='adam', loss='mse')
model2.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0, validation_data=(
pred2 = model2.predict(X_test, verbose=0)
mse2 = mean_squared_error(y_test, pred2)
results.append(['Three Dense Layers', mse2])

# Experiment 3: Dropout after each Dense Layer
model3 = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1)
])
model3.compile(optimizer='adam', loss='mse')
model3.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0, validation_data=(
pred3 = model3.predict(X_test, verbose=0)
mse3 = mean_squared_error(y_test, pred3)
results.append(['Dropout after Dense Layers', mse3])

# Experiment 4: Batch Normalization after Dropout
model4 = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dropout(0.5),
    BatchNormalization(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    BatchNormalization(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    BatchNormalization(),
    Dense(1)
])
model4.compile(optimizer='adam', loss='mse')
model4.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0, validation_data=(
pred4 = model4.predict(X_test, verbose=0)
mse4 = mean_squared_error(y_test, pred4)
results.append(['Batch Norm after Dropout', mse4])

# Create a comparative table
results_df = pd.DataFrame(results, columns=['Experiment', 'MSE'])
results_df
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarni
ng: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequent
ial models, prefer using an `Input(shape)` object as the first layer in the model in
stead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarni
ng: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequent
ial models, prefer using an `Input(shape)` object as the first layer in the model in
stead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarni
ng: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequent
ial models, prefer using an `Input(shape)` object as the first layer in the model in
stead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: UserWarni
ng: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequent
ial models, prefer using an `Input(shape)` object as the first layer in the model in
stead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

Out[ ]:

	Experiment	MSE
0	Single Dense Layer	0.046151
1	Three Dense Layers	0.066092
2	Dropout after Dense Layers	0.151907
3	Batch Norm after Dropout	0.048510

In [ ]: