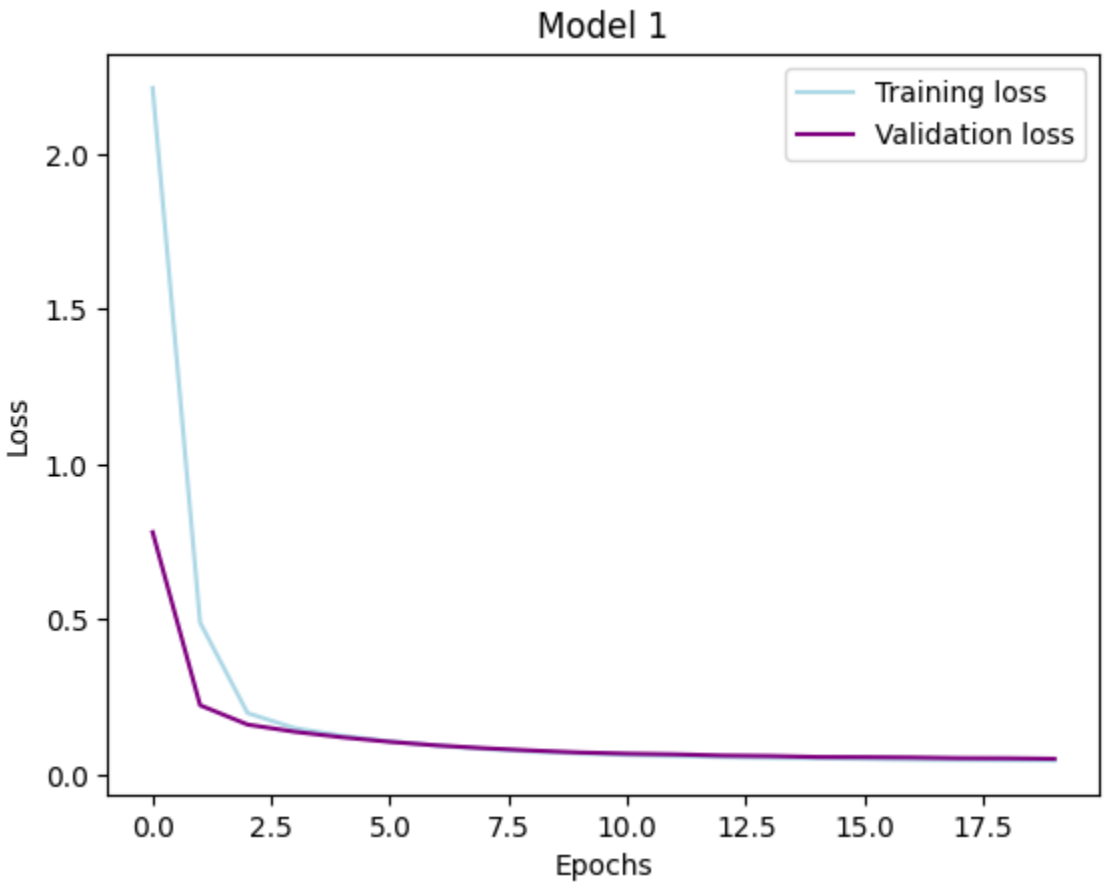
In [1]:
```
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

In [5]:
```
data_path = 'M2_StudentPerformanceData.csv'
student_data = pd.read_csv(data_path)

X = student_data.drop(columns=['StudentID', 'GPA', 'GradeClass'])
y = student_data['GPA']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ran

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

results = {}
```

In [7]:
```
# Definir el modelo
model_1 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(X_train_scaled
    tf.keras.layers.Dense(1)
])

# Compilar el modelo
model_1.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Entrenar el modelo
history_1 = model_1.fit(X_train_scaled, y_train, epochs=20, batch_size=16, v

# Realizar predicciones
y_pred_1 = model_1.predict(X_test_scaled)
mse_1 = mean_squared_error(y_test, y_pred_1)

# Almacenar resultados
results['Model 1'] = mse_1

# Obtener los valores de pérdida y pérdida de validación
loss = history_1.history["loss"]
val_loss = history_1.history["val_loss"]
epochs = range(len(loss))

# Graficar las pérdidas de entrenamiento y validación
plt.figure()
plt.plot(epochs, loss, color="lightblue", label="Training loss")  # color az
plt.plot(epochs, val_loss, color="purple", label="Validation loss")  # color
plt.title("Model 1")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
```

```
plt.show()
```

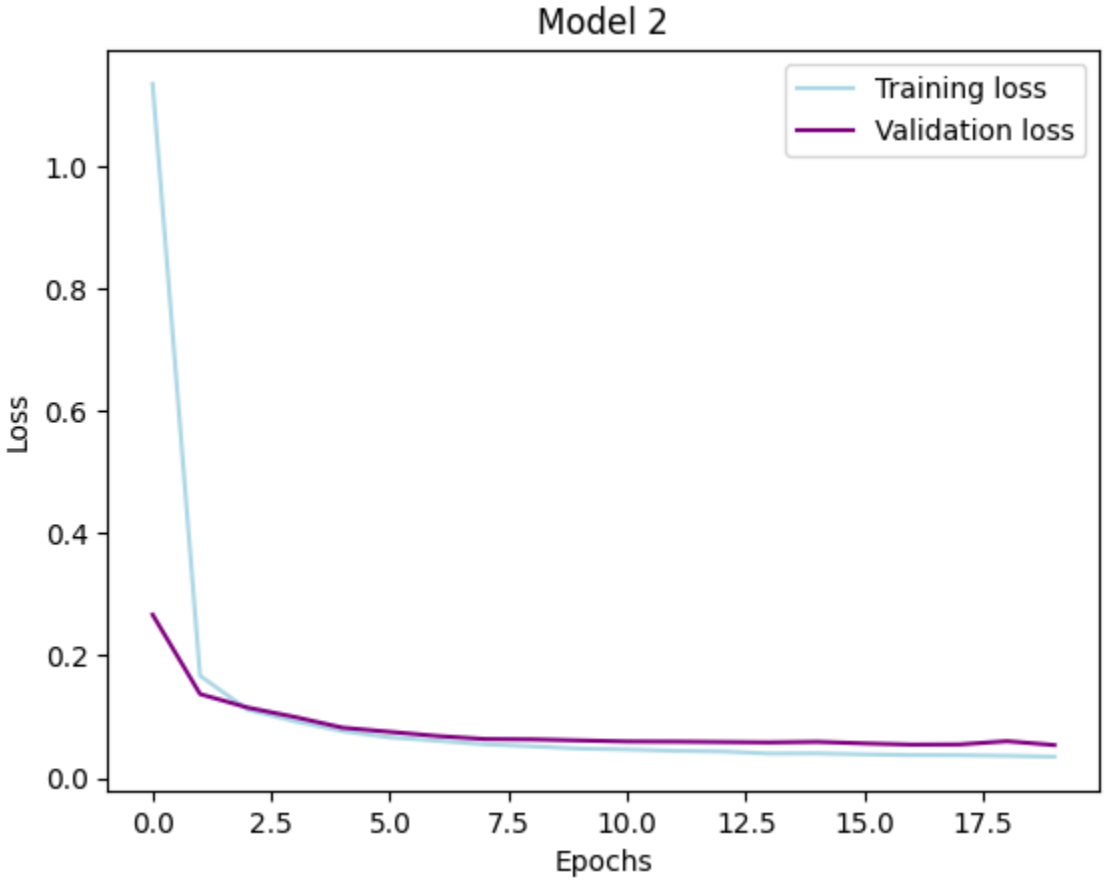
```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: U
serWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Wh
en using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 3ms/step
```



In [8]:
```
# Definir el segundo modelo
model_2 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(X_train_scaled
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dense(1)
])

# Compilar el modelo
model_2.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Entrenar el modelo
history_2 = model_2.fit(X_train_scaled, y_train, epochs=20, batch_size=16, v

# Realizar predicciones
y_pred_2 = model_2.predict(X_test_scaled)
mse_2 = mean_squared_error(y_test, y_pred_2)

# Almacenar resultados
results['Model 2'] = mse_2
```

```
# Obtener los valores de pérdida y pérdida de validación
loss = history_2.history["loss"]
val_loss = history_2.history["val_loss"]
epochs = range(len(loss))

# Graficar las pérdidas de entrenamiento y validación
plt.figure()
plt.plot(epochs, loss, color="lightblue", label="Training loss")  # color az
plt.plot(epochs, val_loss, color="purple", label="Validation loss")  # color
plt.title("Model 2")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

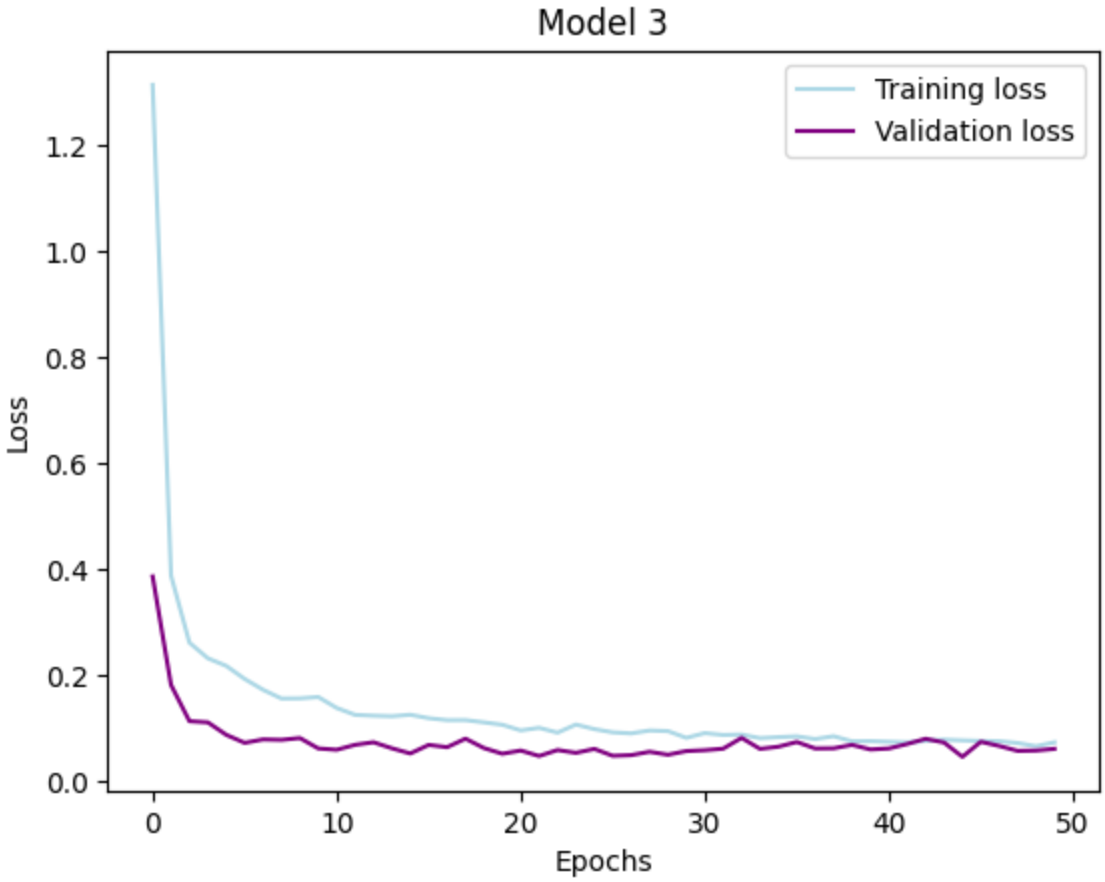
```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: U
serWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Wh
en using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
```



In [9]:
```
# Definir el tercer modelo con capas Dropout
model_3 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(X_train_scaled
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.1),
```

```
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.Dense(1)
])

# Compilar el modelo
model_3.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Entrenar el modelo
history_3 = model_3.fit(X_train_scaled, y_train, epochs=50, batch_size=16, v

# Realizar predicciones
y_pred_3 = model_3.predict(X_test_scaled)
mse_3 = mean_squared_error(y_test, y_pred_3)

# Almacenar resultados
results['Model 3'] = mse_3

# Obtener los valores de pérdida y pérdida de validación
loss = history_3.history["loss"]
val_loss = history_3.history["val_loss"]
epochs = range(len(loss))

# Graficar las pérdidas de entrenamiento y validación
plt.figure()
plt.plot(epochs, loss, color="lightblue", label="Training loss")  # color az
plt.plot(epochs, val_loss, color="purple", label="Validation loss")  # color
plt.title("Model 3")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

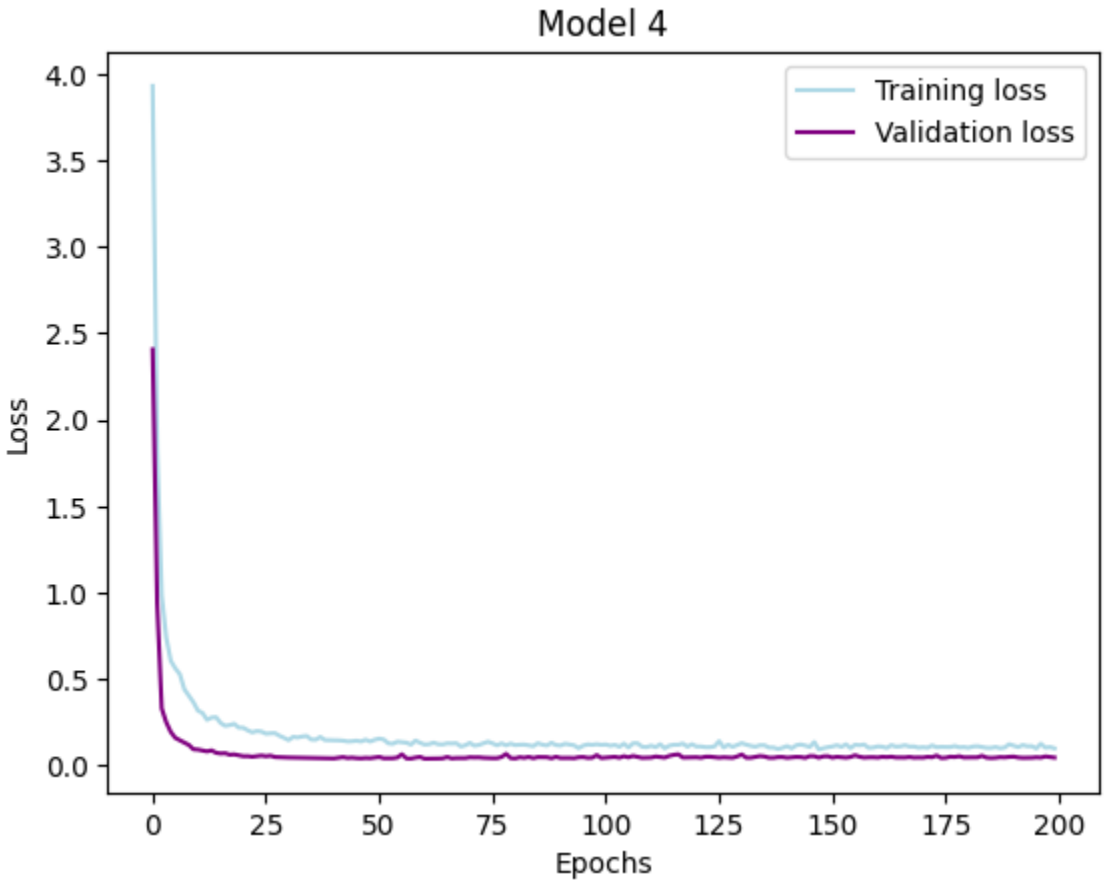
```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: U
serWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Wh
en using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step
```



In [10]:
```python
# Definir el cuarto modelo con capas Dropout y BatchNormalization
model_4 = tf.keras.models.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(X_train_scaled
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.1),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(1)
])

# Compilar el modelo
model_4.compile(optimizer='adam', loss='mse', metrics=['mae'])

# Entrenar el modelo
history_4 = model_4.fit(X_train_scaled, y_train, epochs=200, batch_size=16,

# Realizar predicciones
y_pred_4 = model_4.predict(X_test_scaled)
mse_4 = mean_squared_error(y_test, y_pred_4)

# Almacenar resultados
results['Model 4'] = mse_4

# Obtener los valores de pérdida y pérdida de validación
```

```
loss = history_4.history["loss"]
val_loss = history_4.history["val_loss"]
epochs = range(len(loss))

# Graficar las pérdidas de entrenamiento y validación
plt.figure()
plt.plot(epochs, loss, color="lightblue", label="Training loss")  # color az
plt.plot(epochs, val_loss, color="purple", label="Validation loss")  # color
plt.title("Model 4")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/dense.py:87: U
serWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. Wh
en using Sequential models, prefer using an `Input(shape)` object as the fir
st layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
15/15 ━━━━━━━━━━━━━━━━━━━━ 0s 8ms/step
```

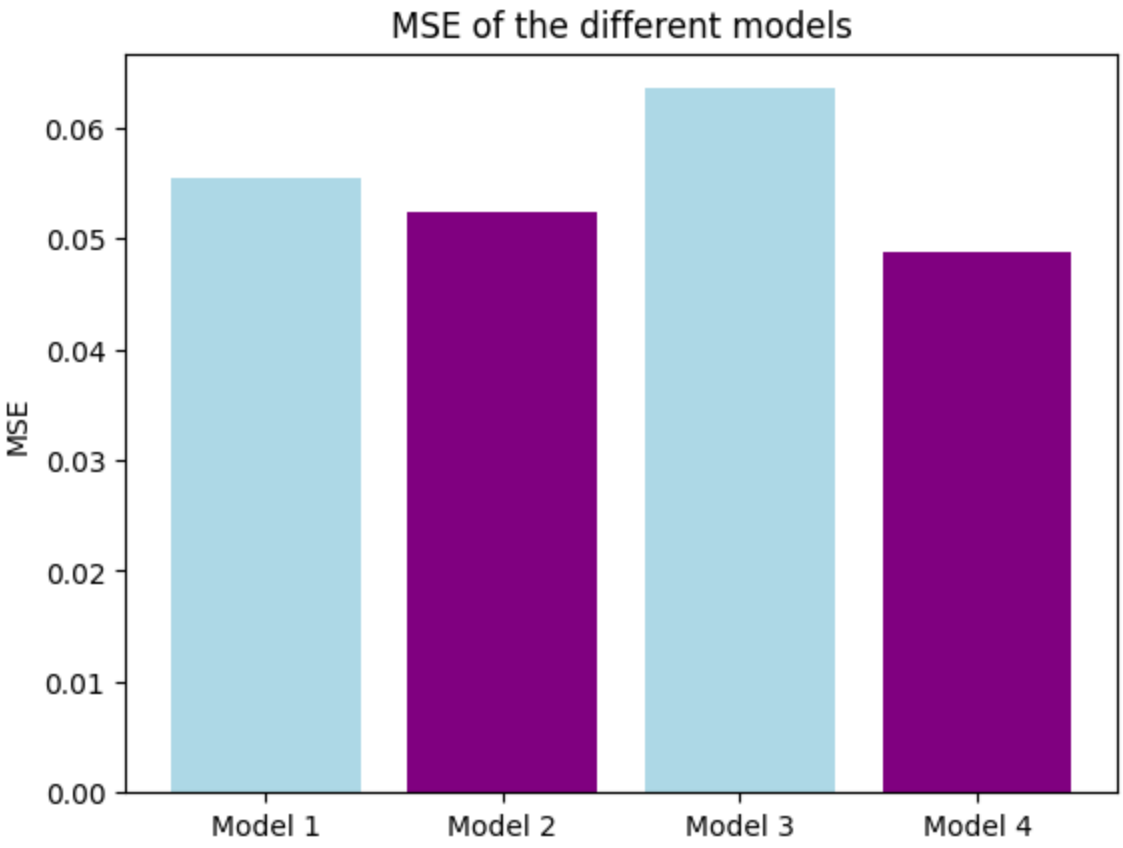


In [11]:
```
results_df = pd.DataFrame(list(results.items()), columns=['Model', 'MSE'])
results_df
```

Out[11]:

	Model	MSE
0	Model 1	0.055504
1	Model 2	0.052467
2	Model 3	0.063528
3	Model 4	0.048834

In [12]:
```
# Graficar los resultados de MSE de los diferentes modelos
plt.figure()
plt.bar(results.keys(), results.values(), color=["lightblue", "purple", "lig
plt.title("MSE of the different models")
plt.ylabel("MSE")
plt.show()
```



Conclusión

El modelo 4, que incluye capas de Dropout y Batch Normalization, obtuvo el mejor desempeño en términos de error cuadrático medio (MSE) en el conjunto de validación, lo que sugiere que estas técnicas ayudaron a mejorar la generalización y estabilidad del modelo. Esto muestra que la inclusión de Dropout y Batch Normalization puede ser efectiva para evitar el sobreajuste y mejorar la precisión en modelos de redes neuronales.

Modelos construidos

- Modelo 1: Una capa densa de 32 neuronas.
- Modelo 2: Tres capas densas de 32 neuronas cada una.
- Modelo 3: Tres capas densas de 32 neuronas cada una, con una capa de Dropout de 0.1 después de cada capa densa.
- Modelo 4: Tres capas densas de 32 neuronas cada una, cada capa densa seguida por una capa de Dropout de 0.1 y una capa de Batch Normalization.