Primero tenemos que saber que tipo de red neuronal y cuantas neuronas y estructura se va a utilizar.

```
model = Sequential()
model.add(Dense(20, input_dim=4, activation='relu'))  # Capa oculta
model.add(Dense(1))  # Capa de salida (sin activación)
```

 Capa de entrada:

- input_dim=4 porque cada muestra de entrada tiene 4 valores:
  [x1, x2, bit_op1, bit_op2]
  (dos números reales y dos bits para codificar la operación)

 Capa oculta:

- Dense(20, activation='relu')
- Tiene 20 neuronas y usa función de activación **ReLU** (rectificada lineal)

 Capa de salida:

- Dense(1)
- Solo una neurona porque la salida es un **número real** (el resultado de la operación)
- No tiene función de activación porque es un **problema de regresión**

En la capa de entrada se reciben 4 números: x1, x2 y 2 bits que en conjunto representan la operación que se desea realizar.

- `00` → suma
- `01` → resta
- `10` → multiplicación
- `11` → división

Y por ejemplo, si recibe de entrada: [5.0, 3.0, 0, 1] → que representa la operación 5 - 3.

```
model.compile(optimizer=Adam(learning_rate=0.01), loss='mse')
```

- **Optimizador:** `Adam`, un algoritmo eficiente de descenso de gradiente
- **Tasa de aprendizaje:** `0.01` (qué tan rápido ajusta los pesos)
- **Función de pérdida:** `mse` (Mean Squared Error)
    - Calcula el error como la diferencia al cuadrado entre el resultado real y el predicho
    - Es común para tareas donde el objetivo es un número continuo

```
model.fit(X_train, y_train, epochs=30, batch_size=64, verbose=2)
```

- Se entrena con **20,000 ejemplos generados aleatoriamente**
- Se ejecuta durante **30 épocas** (pasadas completas por todos los datos)
- **Tamaño de lote (`batch_size`) de 64**, lo que significa que actualiza los pesos cada 64 ejemplos

Esta red neuronal no utiliza un data set de resultados esperados como usted lo había recomendado ya que me pareció que fácilmente el modelo solamente memorizaría. Decidí mejor que los datos de entrenamiento se generen aleatoriamente de 1 a 9

Ya que la red neuronal sea entrenada se puede probar con el siguiente formato:

```
predict_operation(5, 3, 'sub')
```

Esto es un ejemplo de como se ve una prueba de numeros dados por el usuario al modelo: