# Verificar métricas de DL

## Verificar métricas de DL

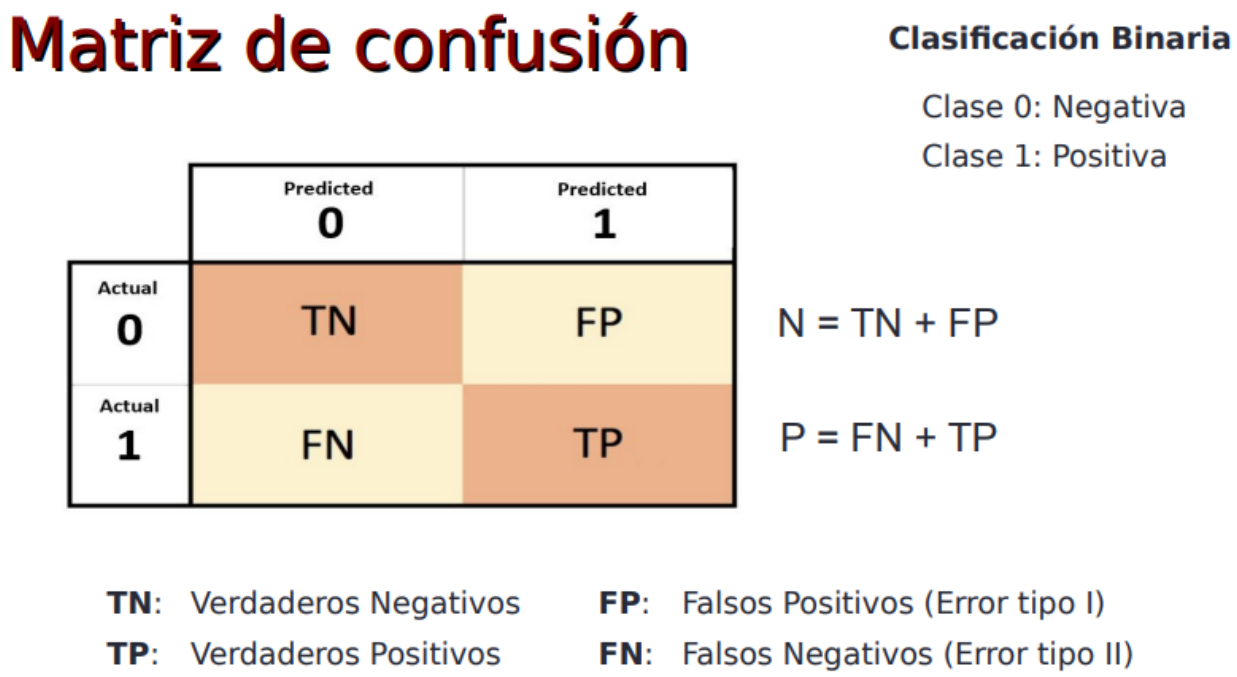
### Objetivo

- Investigar las distintas métricas que se utilizan para verificar si una red neuronal ha logrado identificar apropiadamente los patrones de un conjunto de datos.

### Evaluación de clasificaciones

- precisión: Es la relación entre el número de predicciones correctas y el número total de muestras de entrada. Funciona bien solo si hay igual número de muestras pertenecientes a cada clase.
  $$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$
- Logarithmic Loss: funciona penalizando las clasificaciones falsas. Funciona bien para la clasificación multiclase. El clasificador debe asignar probabilidad a cada clase para todas las muestras.En la formula, **y_ij** indica si la muestra **i** pertenece a la clase **j** o no. **p_ij** indica la probabilidad de que la muestra **i** pertenezca a la clase **j**. Log Loss no tiene límite superior y existe en el rango [0, ∞), lo más cercano a 0 indica una mayor precisión, mientras que si Log Loss está lejos de 0, indica una menor precisión.
  $$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$
- Confusion Matrix: describe el rendimiento completo del modelo.
  - Verdaderos positivos : los casos en los que predijimos SÍ y el resultado real también fue SÍ.
  - Verdaderos negativos : los casos en los que predijimos NO y el resultado real fue NO.
  - Falsos positivos : los casos en los que predijimos SÍ y el resultado real fue NO.

- Falsos negativos : los casos en los que predijimos NO y el resultado real fue SÍ.




- Área bajo la curva (AUC): es el área bajo la curva del gráfico Tasa de falsos positivos frente a Tasa de verdaderos positivos en diferentes puntos en [0, 1] . Cuanto mayor sea el valor, mejor será el rendimiento de nuestro modelo.
  - Tasa de verdaderos positivos (sensibilidad) : la tasa de verdaderos positivos se define como TP/ (FN+TP) . La tasa de verdaderos positivos corresponde a la proporción de puntos de datos positivos que se consideran correctamente como positivos, con respecto a todos los puntos de datos positivos.

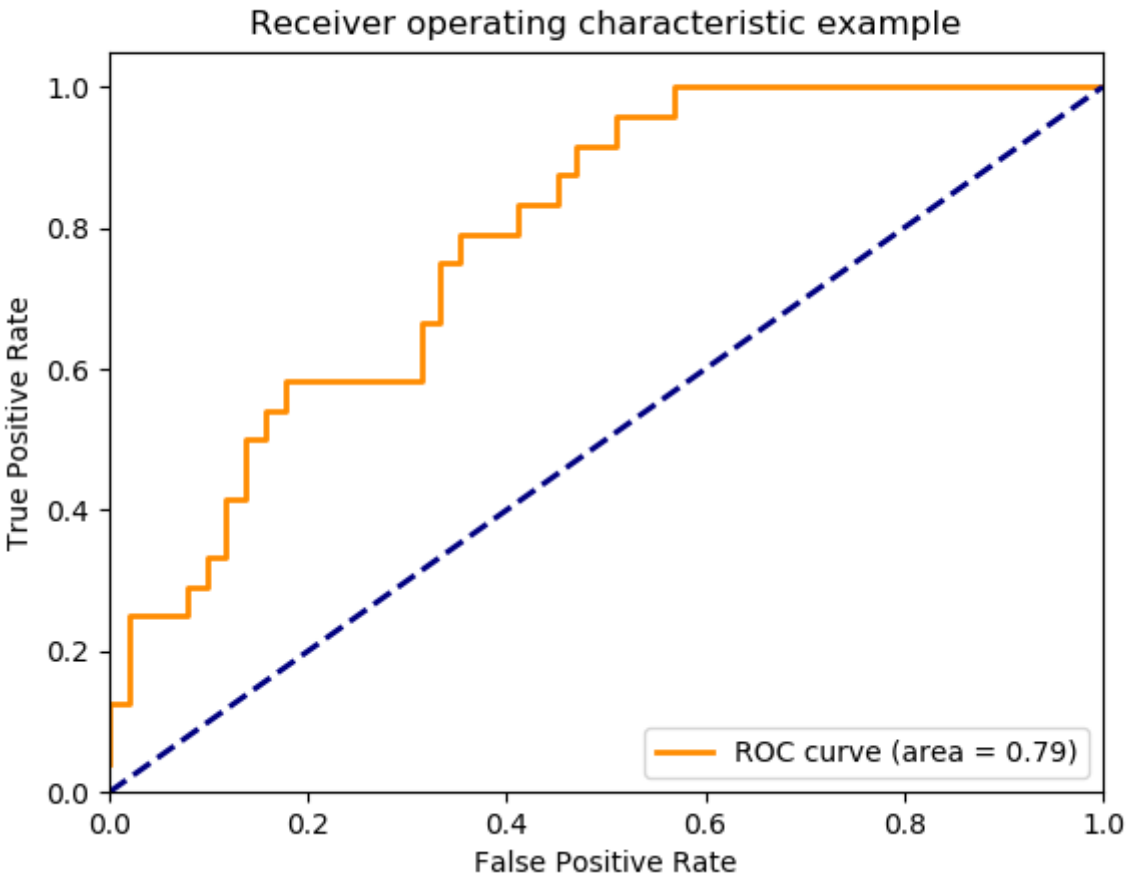
$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

  - Tasa negativa verdadera (especificidad) : la tasa negativa verdadera se define como TN / (FP+TN) . Tasa de falsos positivos corresponde a la proporción de puntos de datos negativos que se consideran correctamente como negativos, con respecto a todos los puntos de datos negativos.

$$TrueNegativeRate = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

  - Tasa de falsos positivos: la tasa de falsos positivos se define como FP / (FP+TN) . La tasa de falsos positivos corresponde a la proporción de puntos de datos negativos que se consideran erróneamente como positivos, con respecto a todos los puntos de datos negativos.

$$FalsePositiveRate = \frac{FalsePositive}{TrueNegative + FalsePositive}$$




- Puntuación F1: es la media armónica entre la precisión y la recuperación. El rango para F1 Score es [0, 1]. Le dice qué tan preciso es su clasificador (cuántas instancias clasifica correctamente), así como qué tan robusto es (no pierde una cantidad significativa de instancias). F1 Score intenta encontrar el equilibrio entre precisión y recuperación.

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

  - Precisión: Es el número de resultados positivos correctos dividido por el número de resultados positivos previstos por el clasificador.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

  - Recall: Es el número de resultados positivos correctos dividido por el número de todas las muestras relevantes (todas las muestras que deberían haber sido identificadas como positivas).

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives}$$

- Error absoluto medio: es el promedio de la diferencia entre los valores originales y los valores pronosticados. Nos da la medida de qué tan lejos estaban las predicciones del resultado real. Sin embargo, no nos dan ninguna idea si estamos prediciendo los datos por debajo o por encima.
- Error cuadrático medio (MSE): toma el promedio del cuadrado de la diferencia entre los valores originales y los valores predichos. La ventaja de MSE es que es más fácil calcular el gradiente, mientras que el error absoluto medio requiere herramientas de programación lineal complicadas

para calcular el gradiente. A medida que tomamos el cuadrado del error, el efecto de los errores más grandes se vuelve más pronunciado que el error más pequeño, por lo tanto, el modelo ahora puede enfocarse más en los errores más grandes.

$$MeanSquaredError = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2$$

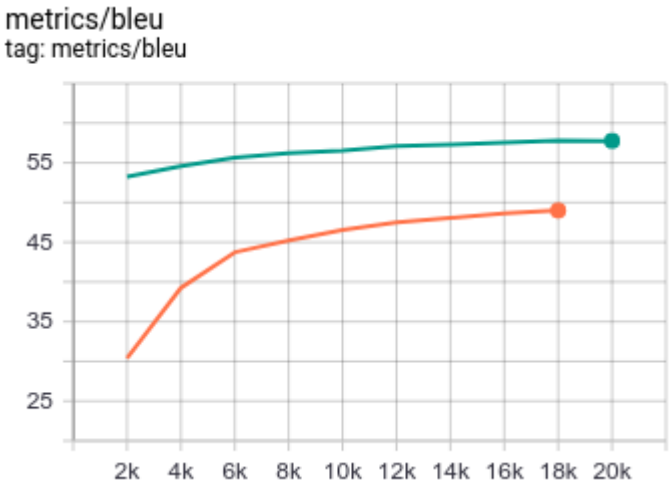
## Evaluación de Redes Neuronales

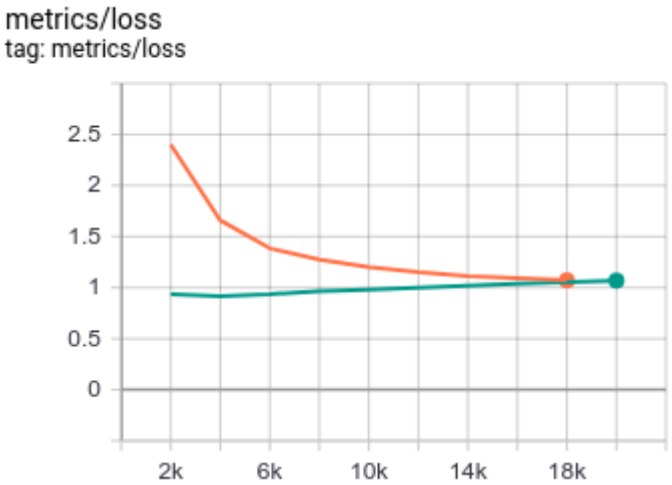
### ¿Qué es una curva de aprendizaje en el aprendizaje automático?

Una de las combinaciones de métricas más utilizadas es la pérdida de entrenamiento + pérdida de validación a lo largo del tiempo.

La pérdida de entrenamiento indica qué tan bien se ajusta el modelo a los datos de entrenamiento, mientras que la pérdida de validación indica qué tan bien se ajusta el modelo a los datos nuevos.


- Curvas de aprendizaje de optimización : curvas de aprendizaje calculadas sobre la métrica por la cual se optimizan los parámetros del modelo, como pérdida o error cuadrático medio.
- Tasa de verdaderos positivos (sensibilidad) : <se define como TP/ (FN+TP) . La tasa de verdaderos positivos corresponde a la proporción de puntos de datos positivos que se consideran correctamente como positivos, con respecto a todos los puntos de datos positivos.
- Curvas de aprendizaje de rendimiento : curvas de aprendizaje calculadas en la métrica por la cual se evaluará y seleccionará el modelo, como exactitud, precisión, recuperación o puntaje F1.



Podemos detectar problemas en el comportamiento de un modelo observando la evolución de una curva de aprendizaje.

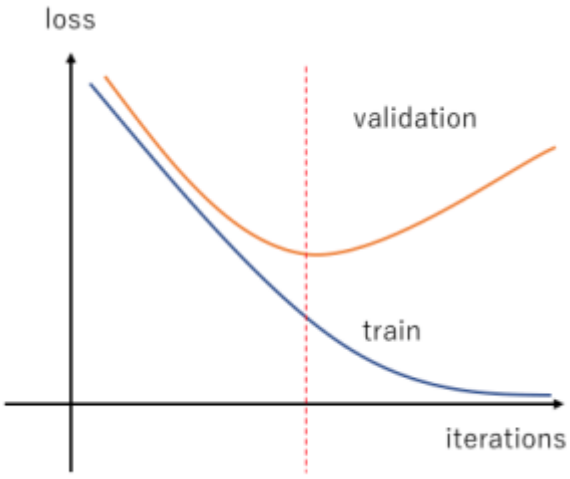
- Sesgo : se produce un alto sesgo cuando el algoritmo de aprendizaje no tiene en cuenta toda la información relevante, por lo que no puede capturar la riqueza y complejidad del modelo.
- Underfitting : cuando el algoritmo no es capaz de modelar ni los datos de entrenamiento ni los nuevos datos, obteniendo constantemente valores de error elevados que no disminuyen con el tiempo.

¿cómo podemos usar las curvas de aprendizaje para detectar que nuestro modelo no se ajusta bien?


La función de costo (pérdida) es alta y no decrece con el número de iteraciones, tanto para las curvas de validación como de entrenamiento.
De hecho, podríamos usar solo la curva de entrenamiento y verificar que la pérdida sea alta y que no disminuya, para ver que no se ajusta bien.

### Alta varianza/sobreajuste

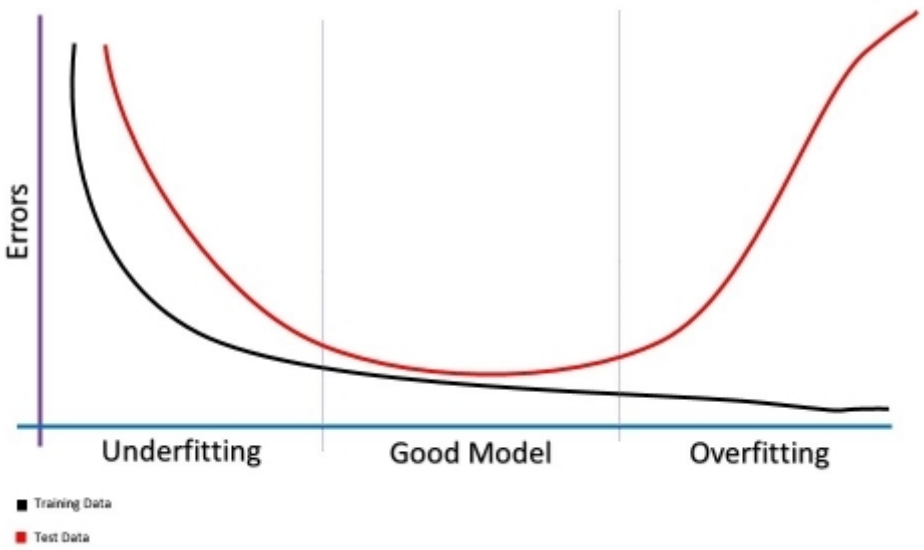
- Varianza : la varianza alta ocurre cuando el modelo es demasiado complejo y no representa los patrones reales más simples que existen en los datos.
- Sobreajuste : el algoritmo captura bien los datos de entrenamiento, pero se desempeña mal con los datos nuevos, por lo que no puede generalizar.



- La pérdida de entrenamiento va disminuyendo con el tiempo, consiguiendo valores de error bajos
- La pérdida de validación desciende hasta que se encuentra un punto de inflexión, y allí comienza a subir de nuevo. Ese punto representa el comienzo del sobreajuste.

### Encontrar la compensación correcta de sesgo/varianza

Podemos usar las curvas de pérdida de entrenamiento y validación para encontrar la compensación correcta de sesgo/varianza:




- El proceso de entrenamiento debe detenerse cuando la tendencia del error de validación cambia de descendente a ascendente.
- Si detenemos el proceso antes de ese punto, el modelo fallará
- Si detenemos el proceso después de ese punto, el modelo se sobreajustará

### Validando el entrenamiento

Para monitorear durante el entrenamiento la precisión del modelo en datos que nunca antes había visto, creará un conjunto de validación separando 10,000 muestras de los datos de entrenamiento originales.

¿Cómo podríamos usar las curvas de aprendizaje para detectar que un modelo se está sobreajustando?

```
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

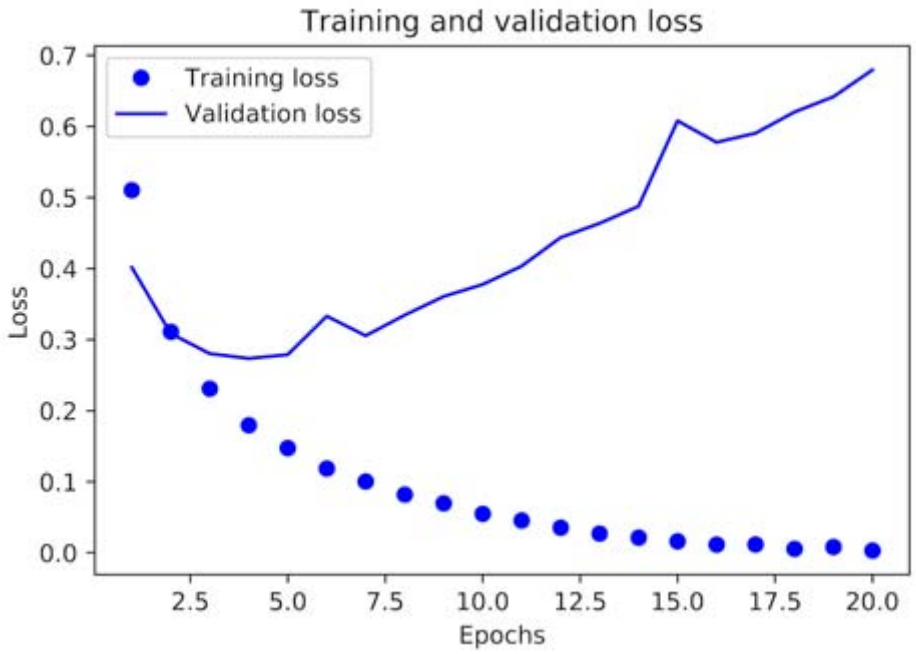
Ahora entrenará el modelo durante 20 épocas (20 iteraciones sobre todas las muestras en los tensores x_train y y_train), en mini lotes de 512 muestras.
Al mismo tiempo, controlará la pérdida y la precisión de las 10 000 muestras que separó.
Lo hace pasando los datos de validación como el argumento validation_data.

```
model.compile(optimizer='rmsprop',
loss='binary_crossentropy',
metrics=['acc'])
history = model.fit(partial_x_train,
partial_y_train,
epochs=20,
batch_size=512,
validation_data=(x_val, y_val))
```

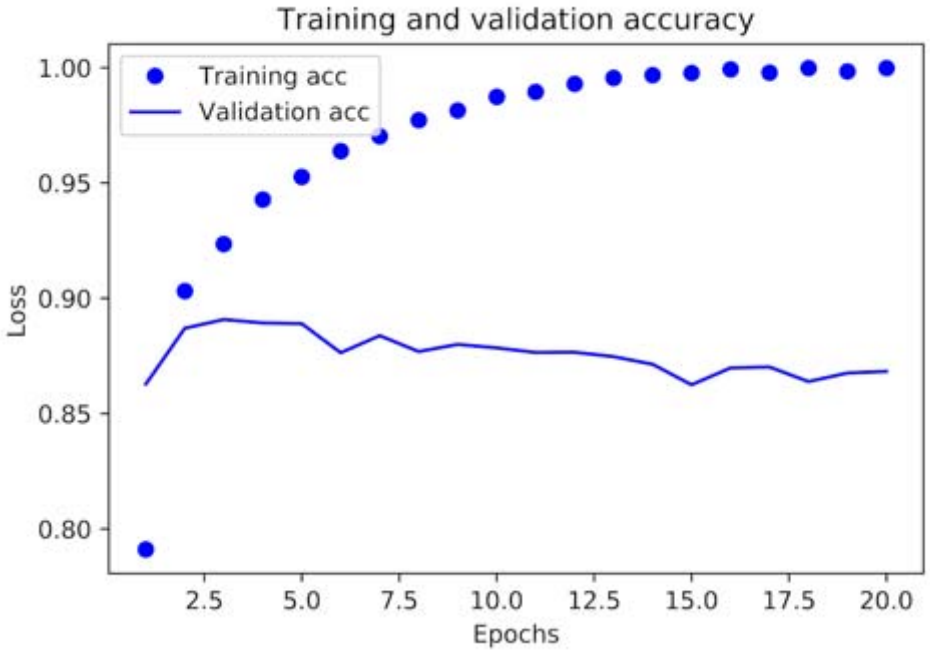
Al final de cada época, hay una pequeña pausa mientras el modelo calcula su pérdida y precisión en las 10 000 muestras de los datos de validación. Tenga en cuenta que la llamada a model.fit() devuelve un objeto Historial. Este objeto tiene un historial de miembros, que es un diccionario que contiene datos sobre todo lo que sucedió durante el entrenamiento.

El diccionario contiene cuatro entradas: una por métrica que se estaba monitoreando durante el entrenamiento y durante la validación.
En las siguientes dos listas se utilizan para trazar la pérdida de entrenamiento y validación una al lado de la otra, así como la precisión del entrenamiento y la validación.

```
import matplotlib.pyplot as plt
history_dict = history.history
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, loss_values, 'bo', label='Training loss')
plt.plot(epochs, val_loss_values, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
plt.clf()
Clears the figure
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




**Conclusiones**

- La pérdida de entrenamiento disminuye con cada época y la precisión del entrenamiento aumenta con cada época. Eso es lo que esperaría al ejecutar la optimización de descenso de gradiente: la cantidad que está tratando de minimizar debería ser menor con cada iteración.
  Pero ese no es el caso de la pérdida de validación y la precisión: parecen alcanzar su punto máximo en la cuarta época. Un modelo que funciona mejor con los datos de entrenamiento no es necesariamente un modelo que funcionará mejor con datos que nunca antes había visto.
- En términos precisos, lo que está viendo es un sobreajuste: después de la segunda época, está sobreoptimizando los datos de entrenamiento y termina aprendiendo representaciones que son específicas de los datos de entrenamiento y no se generalizan con datos fuera del entrenamiento.
- En este caso, para evitar el sobreajuste, podría dejar de entrenar después de tres épocas. En general, puede usar una variedad de técnicas para mitigar el sobreajuste. Para mejorarlo entrenemos una nueva red desde cero durante cuatro épocas y luego la evaluaremos en los datos de prueba.