Tópicos en Implementación: Aprendizaje Profundo

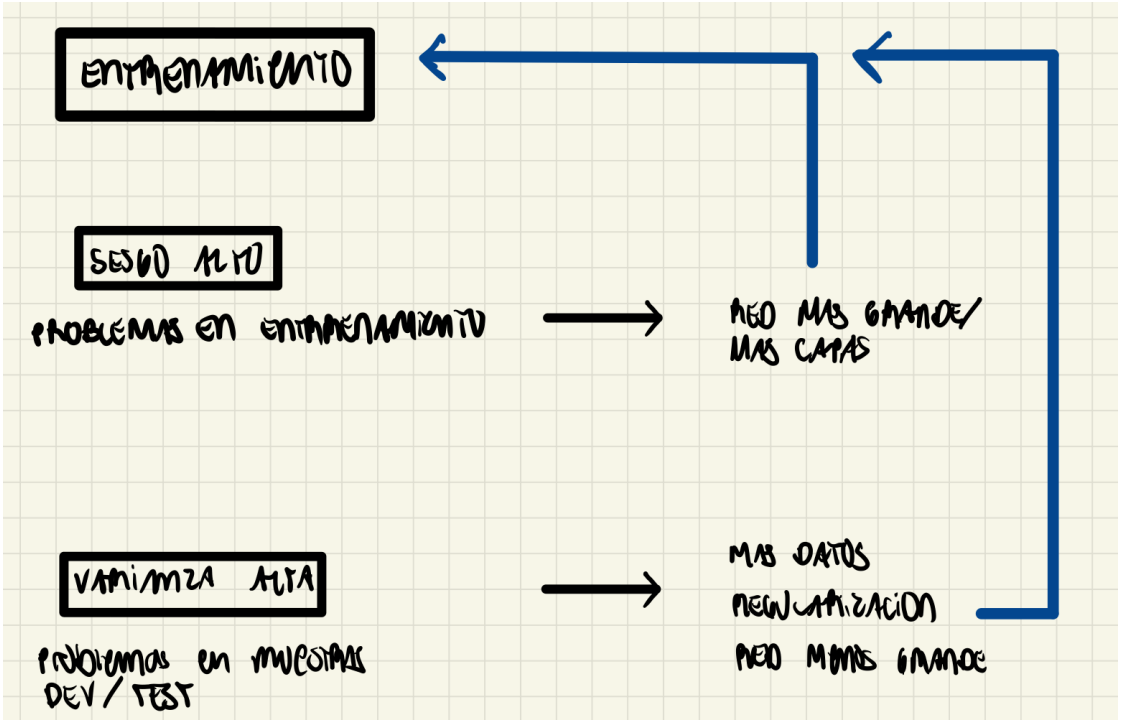
Marzo de 2026

Contenido

Entrenamiento: Datos

- Las muestras de entrenamiento son cada vez más pequeñas.
- Tradicionalmente una muestra se separaba en 70 % entrenamiento (desarrollo), 15 % validación y 15 % prueba.
- Con la gran cantidad de información disponible actualmente ahora es más común una muestra de 96 % entrenamiento (desarrollo), 2 % validación y 2 % prueba.

Entrenamiento: Análisis



Entrenamiento: Sesgo vs. Varianza

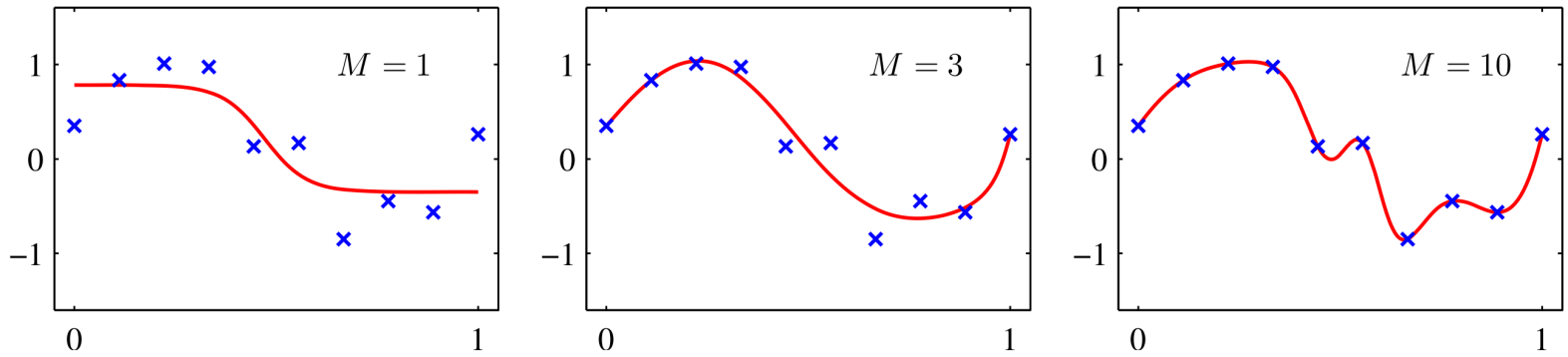
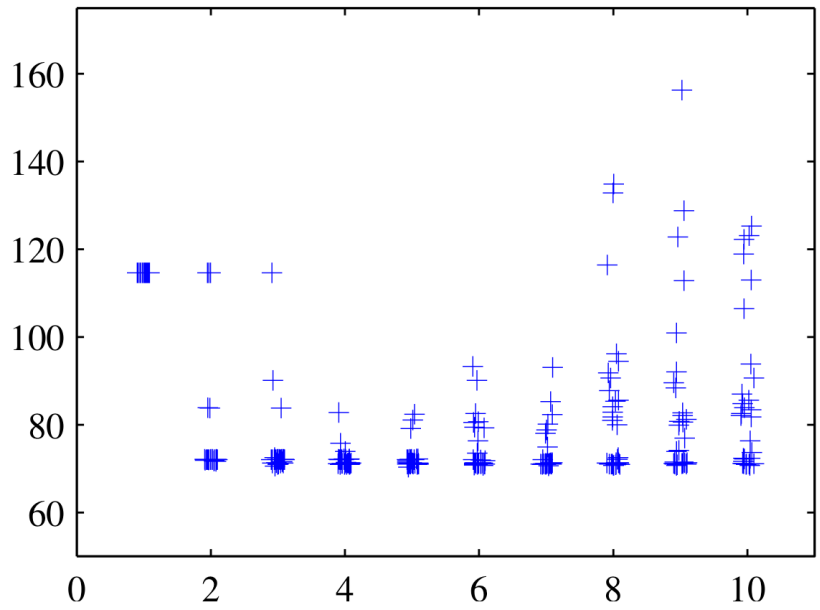


Figure 5.9 Examples of two-layer networks trained on 10 data points drawn from the sinusoidal data set. The graphs show the result of fitting networks having $M = 1$, 3 and 10 hidden units, respectively, by minimizing a sum-of-squares error function using a scaled conjugate-gradient algorithm.

Entrenamiento: Óptimos Locales

Figure 5.10 Plot of the sum-of-squares test-set error for the polynomial data set versus the number of hidden units in the network, with 30 random starts for each network size, showing the effect of local minima. For each new start, the weight vector was initialized by sampling from an isotropic Gaussian distribution having a mean of zero and a variance of 10.



Entrenamiento: Data Augmentation

- Para aumentar la cantidad de datos, variedad, etc. se puede usar: rotaciones, traslaciones, reflexiones.
- Acercamientos.
- Agregar ruido.

Regularización: Weight Decay

- La función de costo se define como:

$$J_R(w^{[1]}, b^{[1]}, ..., w^{[L]}, b^{[L]}) = J(w^{[1]}, b^{[1]}, ..., w^{[L]}, b^{[L]}) \quad (1)$$

$$+\frac{\lambda}{2m}\sum_{l=1}^{L}\|w^{[l]}\|_F^2 \quad (2)$$

donde J es la función de costo sin regularización.

- Luego, para hacer backpropagation basta con reemplazar el gradiente por:

$$dw_R^{[l]} \leftarrow dw^{[l]} + \frac{\lambda}{m}w^{[l]} \quad (3)$$

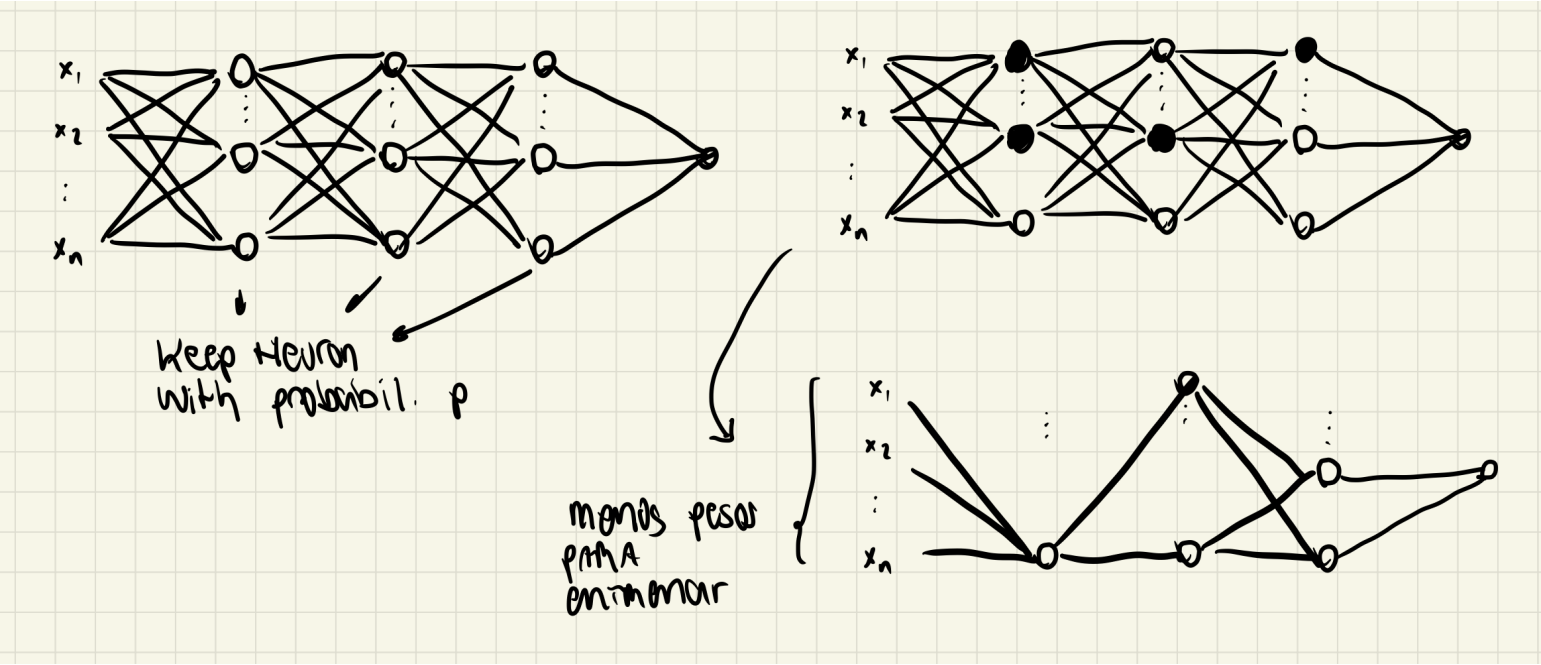
$$= (1 - \frac{\alpha\lambda}{m})w^{[l]} - \alpha dw^{[l]} \quad (4)$$

donde $dw^{[}l]$ es el gradiente sin regularización y el término $(1 - \frac{\alpha\lambda}{m})$ justifica el nombre de weight decay.

Regularización: Weight Decay

- La regularización controla la complejidad del modelo.
- Intuitivamente: si la penalidad aumenta, los pesos se hacen pequeños lo que hace que $\|z\|$ se vuelva pequeño y las funciones de activación sean aproximadamente lineales.

Regularización: Dropout



Regularización: Dropout

- En el momento de implementar dropout es importante reescalar las funciones de activación porque si se reducen p las neuronas (funciones de activación) entonces z se construye con una proporción p menos de términos:

$$z^{[l]} = w^{[l]} a^{[l]} + b^{[l]} \tag{5}$$

- Luego, se propone reesalar asi:

$$z^{[l]} = \frac{w^{[l]} a^{[l]}}{p} + b^{[l]} \tag{6}$$

Regularización: Dropout

- Dropout distribuye la importancia de las neuronas impidiendo que una tenga demasiado peso.
- En principio cada capa puede tener una probabilidad distinta de dropout.
- Cuando se hacen prediciones se usa la red entrenada, no se aleatoriza.
- El entrenamiento puede tardar un poco más. Las actualizaciones son ruidosas.
- Como la función de costos no decrece monotonicamnte, es una buena práctca primero verificar entrenamiento sin dropout.

Regularización: Early Stopping

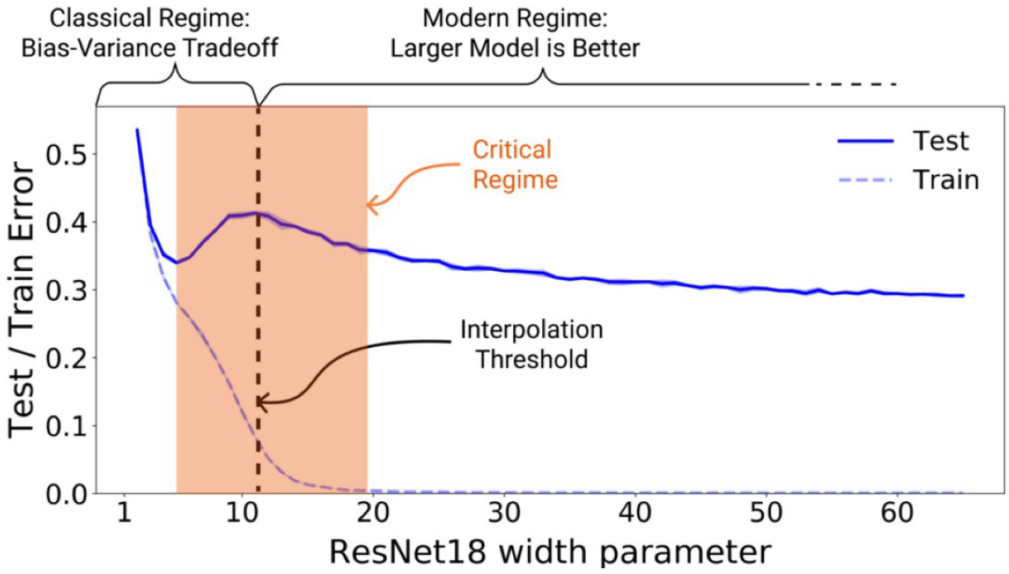


Figure 9.9 Plot of training set and test set errors for a large neural network model called ResNet18 trained on an image classification problem versus the complexity of a model. The horizontal axis represents a hyperparameter governing the number of hidden units and hence the overall number of weights and biases in the network. The vertical dashed line, labelled 'interpolation threshold' indicates the level of model complexity at which the model is capable, in principle, of achieving zero error on the training set. [From Nakkiran *et al.* (2019) with permission.]

- Al comienzo de la franja naranja sería ideal hacer *early stopping*. El problema es que es una estrategía que mezcla dos problemas: la minimización de la función de costos y evitar el sobre ajuste del modelo.
- Una alternativa computacionalmente costosa es utilizar una muestra de validación.

Double Descent como función de la complejidad

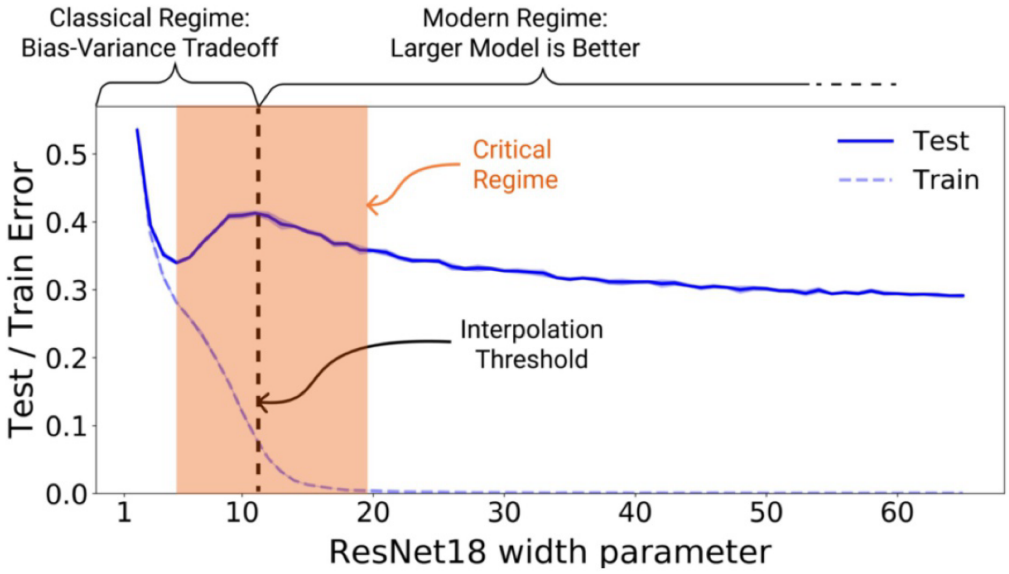


Figure 9.9 Plot of training set and test set errors for a large neural network model called ResNet18 trained on an image classification problem versus the complexity of a model. The horizontal axis represents a hyperparameter governing the number of hidden units and hence the overall number of weights and biases in the network. The vertical dashed line, labelled 'interpolation threshold' indicates the level of model complexity at which the model is capable, in principle, of achieving zero error on the training set. [From Nakkiran *et al.* (2019) with permission.]

- Al comienzo de la franja naranja sería ideal hacer *early stopping*. El problema es que es una estrategía que mezcla dos problemas: la minimización de la función de costos y evitar el sobre ajuste del modelo.
- Una alternativa computacionalmente costosa es utilizar una muestra de validación.

Double Descent como función de las epocas

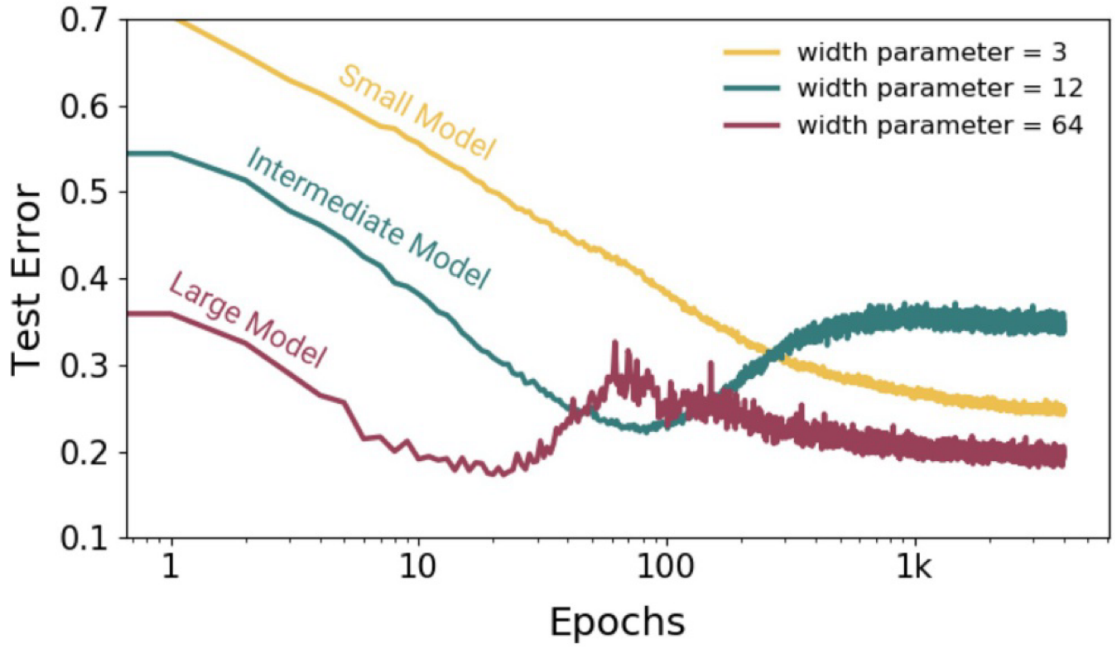


Figure 9.10 Plot of test set error versus number of epochs of gradient descent training for ResNet18 models of various sizes. The effective model complexity increases with the number of training epochs, and the double descent phenomenon is observed for a sufficiently large model. [From Nakkiran *et al.* (2019) with permission.]

Double Descent como función de ejemplos

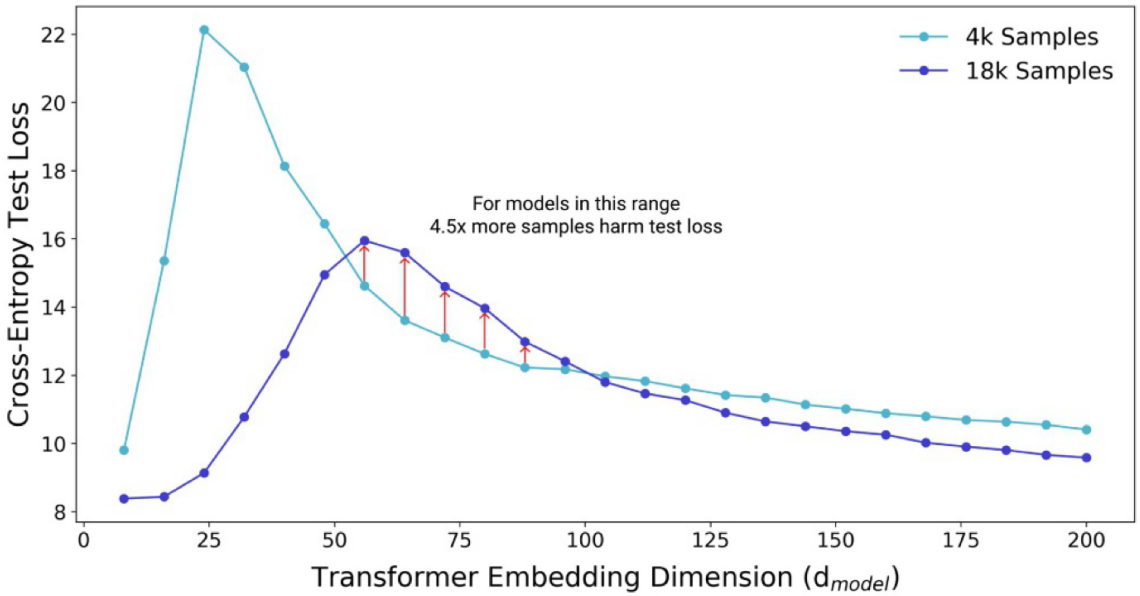


Figure 9.11 Plot of test set error for a large transformer model versus the embedding dimension, which controls the number of parameters in the model. Increasing the size of the training set from 4,000 to 18,000 samples generally leads to a lower test set error, but for some intermediate values of model complexity, there can be an increase in the error, as shown by the vertical red arrows. [From Nakkiran *et al.* (2019) with permission.]

Inductive Bias

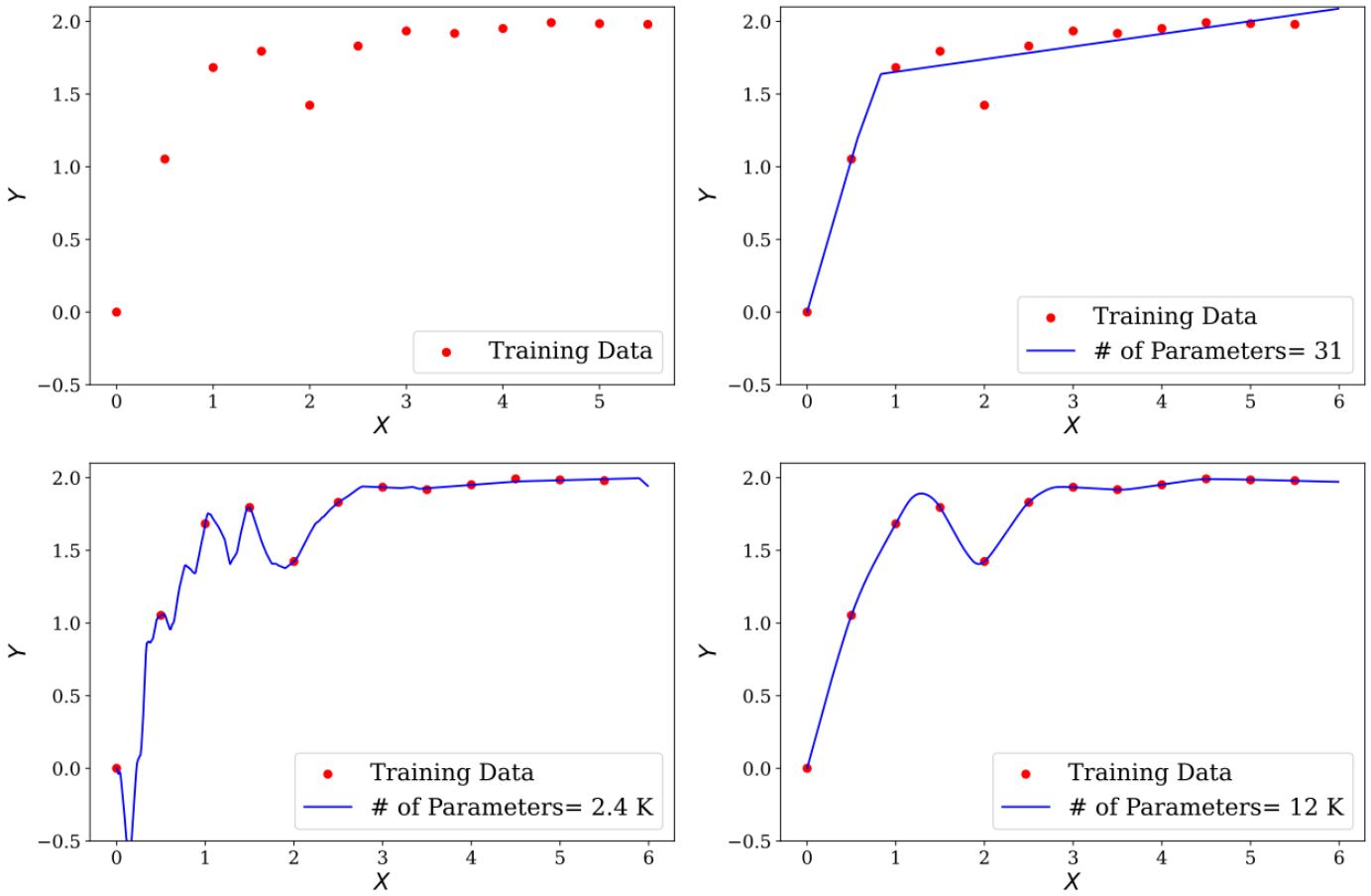


Figure 13: Interpolation with an overparameterized deep neural network.