https://www.sli.do/
#073374

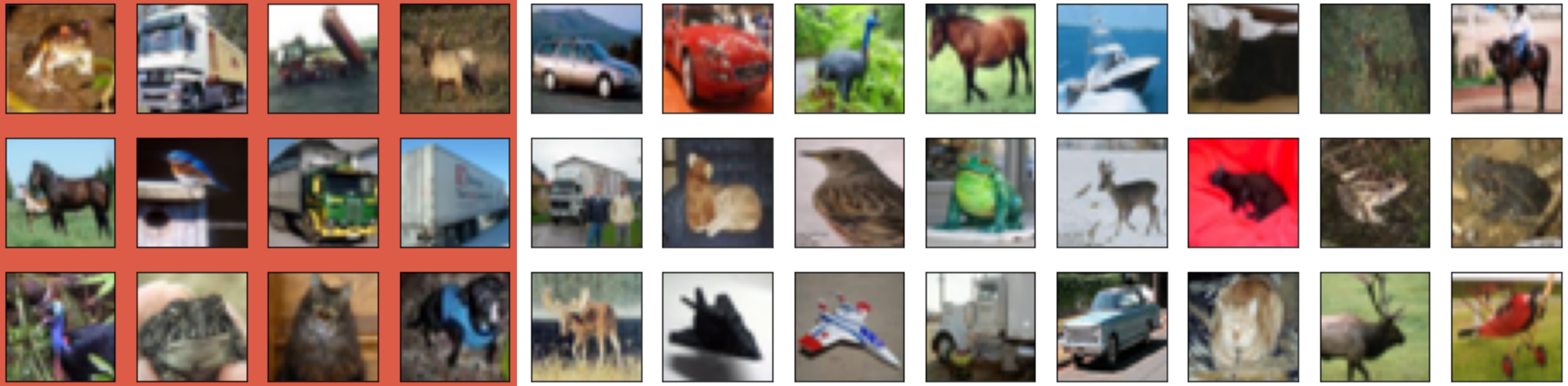
Deep Learning for Computer Vision (II)

Learning Objectives

- Learn a simple end-to-end image classification using CNNs.
- Learn the basic concepts to fine tune the hyperparameters of CNNs.

A simple end-to-end image classification using CNNs

- Construct a simple CNN to classify images from the CIFAR-10 dataset
- CIFAR-10 consists of 60,000 (32 × 32) color images containing 1 of 10 object classes, with 6,000 images per class.
- We will walk you through some common process on using CNN.
- See CNN_Cifar10_Simple.pdf



Improve Performance of Your CNN Model

- Network architecture
  - Number of hidden layers (network depth)
  - Number of neurons in each layer (layer width)
  - Activation type
- Learning and optimization
  - Learning rate and decay schedule
  - Mini-batch size
  - Optimization algorithms
  - Number of training iterations or epochs (and early stopping criteria)
- Regularization techniques to avoid overfitting
  - L2 regularization
  - Dropout layers
  - Data augmentation
- Batch normalization
- Transfer learning

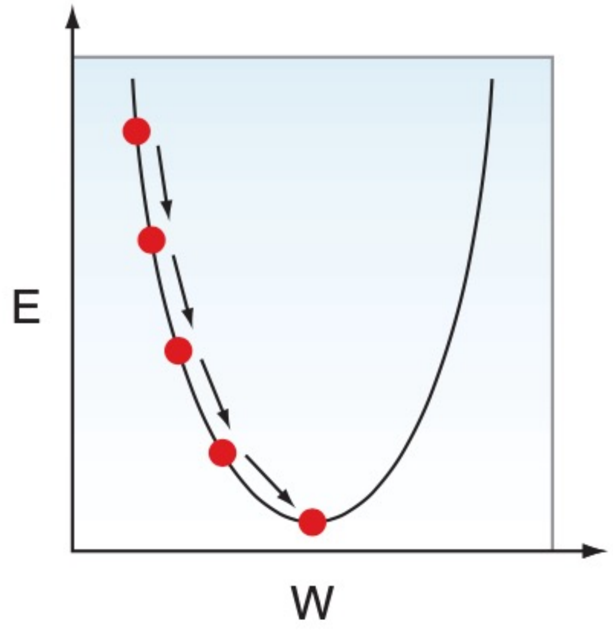
Learning Rate

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \cdot \nabla E\big(\mathbf{w}(t)\big)$$

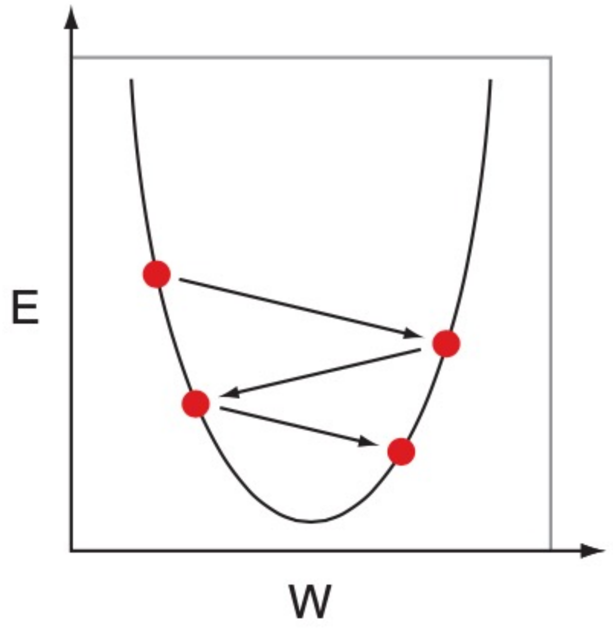
$0 < \eta < 1$

The amount that the weights are updated during training

A learning rate *smaller* than the ideal value: the model takes smaller steps down the error curve.



A learning rate *larger* than the ideal value: the optimizer overshoot the optimal weight value.

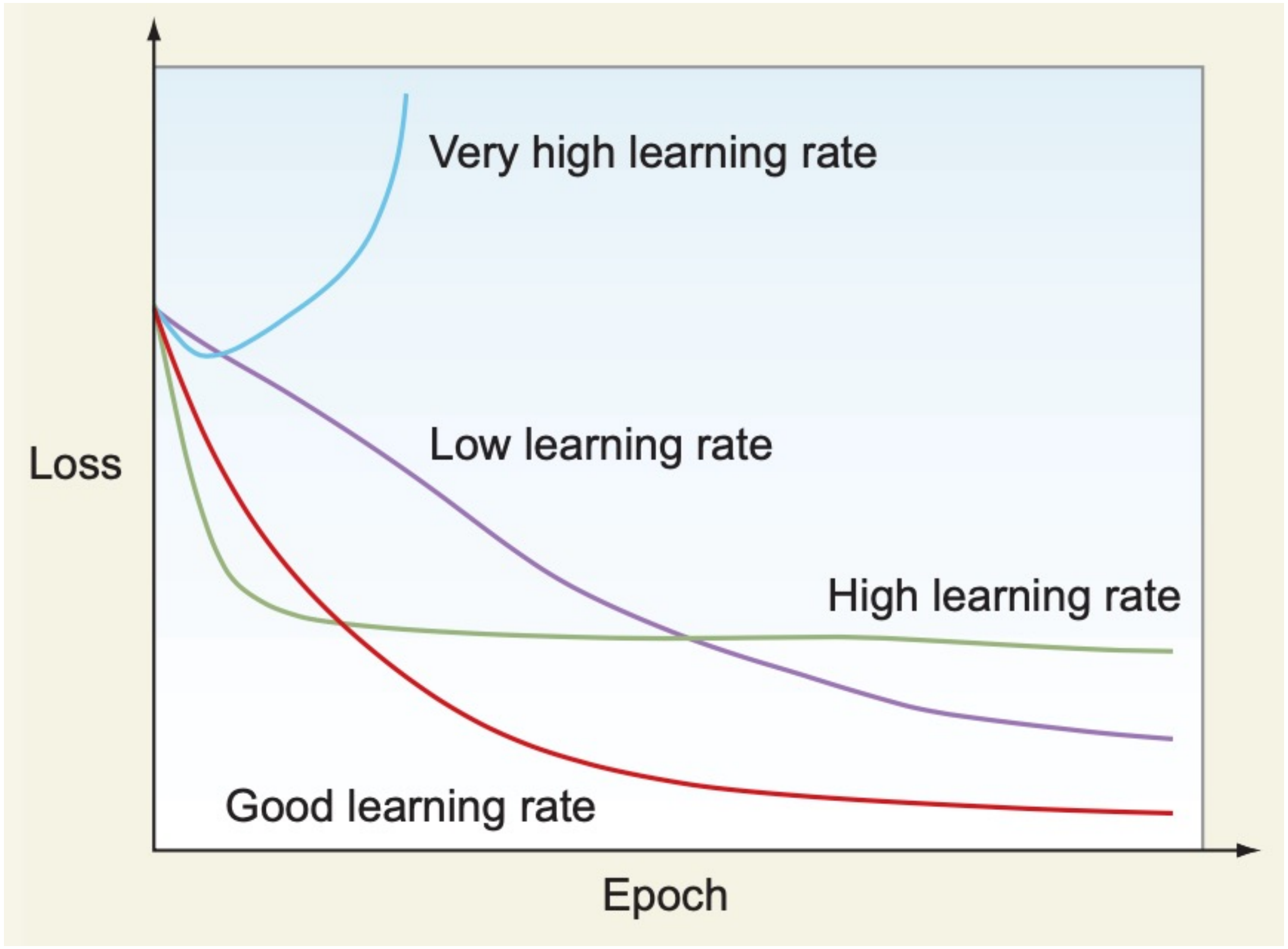


Learning Rate

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \cdot \nabla E\big(\mathbf{w}(t)\big)$$

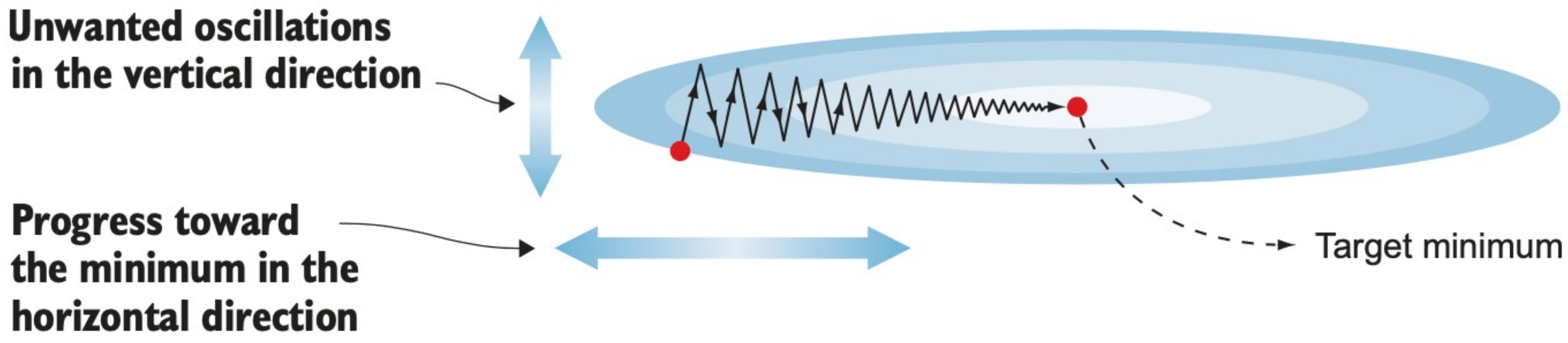
$0 < \eta < 1$

The amount that the weights are updated during training



Optimization Algorithms: Gradient Descent with Momentum

- Larger learning rate in GD often leads to oscillation behavior and slow down the convergence


Unwanted oscillations in the vertical direction
Progress toward the minimum in the horizontal direction
Target minimum

- Momentum: lets the GD navigate along relevant directions and softens the oscillation in irrelevant directions.

$$\mathbf{w}(t+1) = \mathbf{w}(t) - \eta \cdot \nabla E\big(\mathbf{w}(t)\big) + \textbf{velocity term}$$

Optimization Algorithms: Adaptive Learning Rate

- The learning rate adapts over time on the basis of historical values of the gradient.
- **AdaGrad** and **RMSProp**

tf.keras.optimizers.Adagrad(learning_rate=0.001, initial_accumulator_value=0.1)

tf.keras.optimizers.RMSprop(learning_rate=0.001, rho=0.9)

- **Adam**: combine adaptive learning rate and momentum.

tf.keras.optimizers.Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999)

Early Stopping

- Early stopping is an algorithm widely used to determine the right time to stop the training process before overfitting happens. It simply monitors the validation error value and stops the training when the value starts to increase.

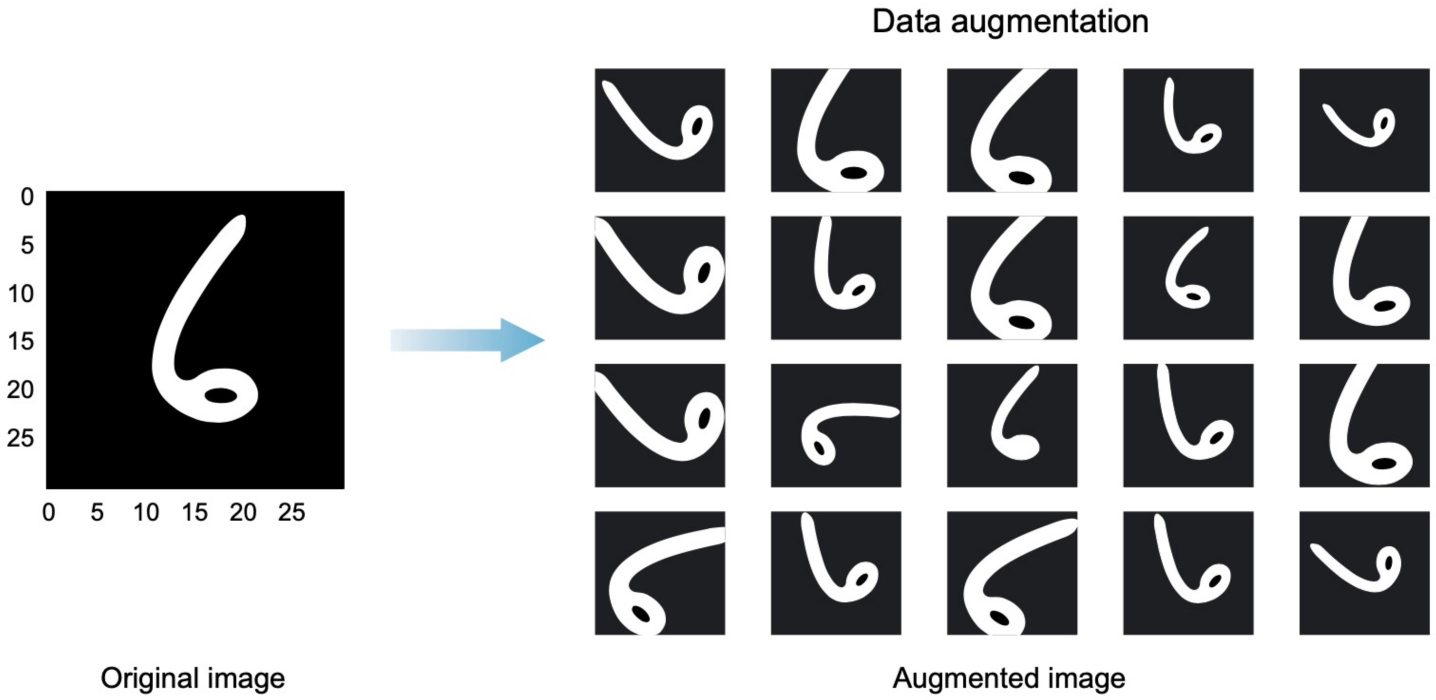
- CNN_Cifar10_Simple_EarlyStop.ipynb

Weight Regularization and Dropout

- Regularization techniques are commonly used to fight overfitting.
- Typical weight regularization techniques include L1 and L2 regularization.
- Dropout is another regularization technique that is very effective for simplifying a neural network and avoiding overfitting.
- CNN_Regularization.pdf

Data Augmentation

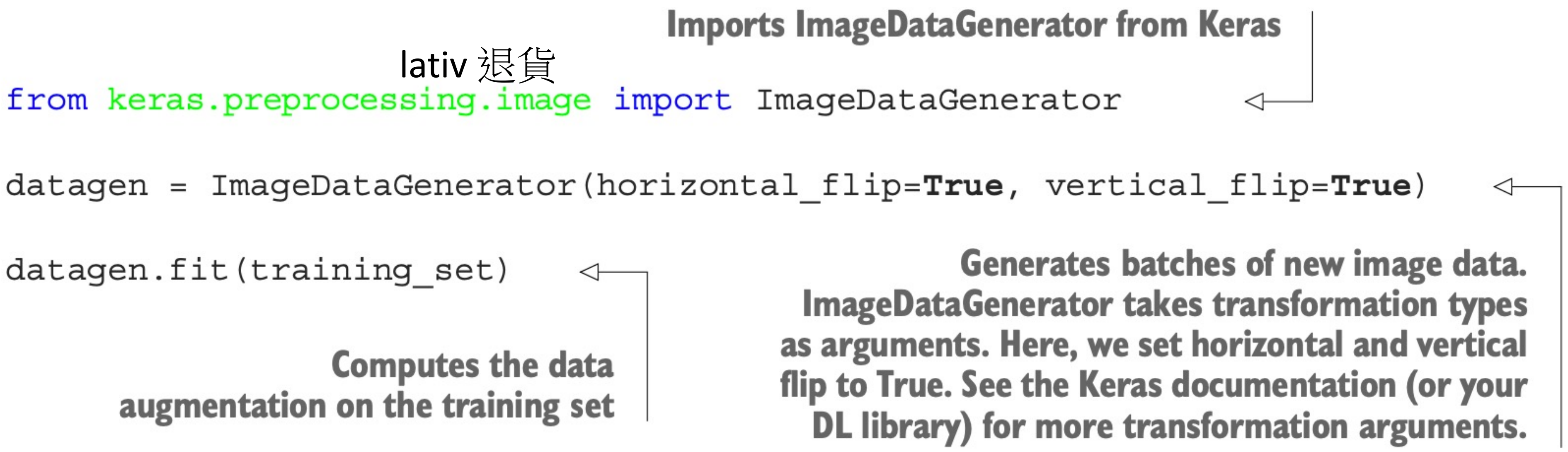
- One way to avoid overfitting is to obtain more data. Since this is not always a feasible option, we can **augment** our training data by generating new instances of the same images with some **transformations**.
- **Data augmentation** can be an inexpensive way to give your learning algorithm more training data and therefore reduce overfitting.



Data Augmentation

- One way to avoid overfitting is to obtain more data. Since this is not always a feasible option, we can **augment** our training data by generating new instances of the same images with some **transformations**.
- **Data augmentation** can be an inexpensive way to give your learning algorithm more training data and therefore reduce overfitting.

lativ 退貨



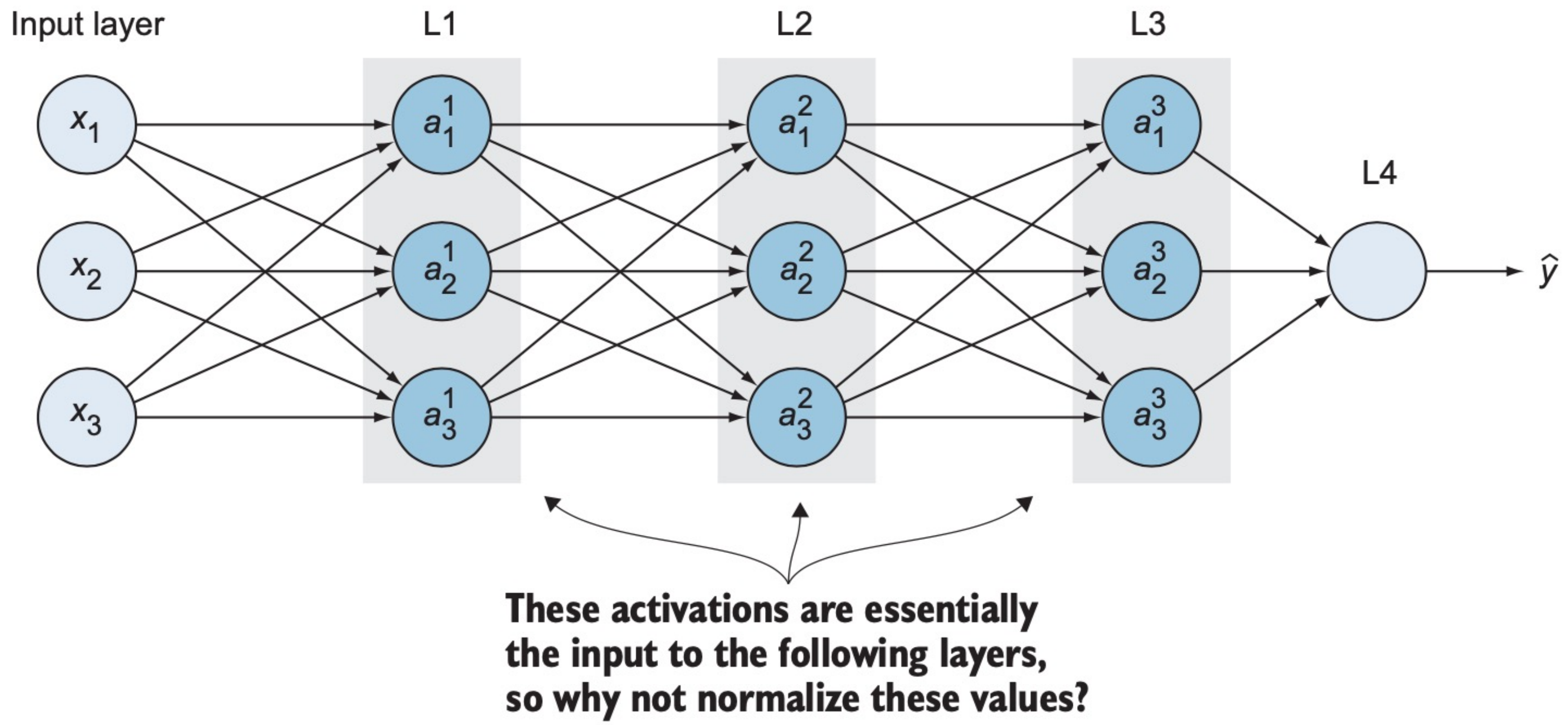
Imports ImageDataGenerator from Keras

```
from keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)

datagen.fit(training_set)
```

Generates batches of new image data. ImageDataGenerator takes transformation types as arguments. Here, we set horizontal and vertical flip to True. See the Keras documentation (or your DL library) for more transformation arguments.

Computes the data augmentation on the training set

Batch Normalization



- This process is called **batch normalization** (BN) and is used liberally in many of the advanced convnet architectures that come packaged with Keras, such as ResNet50, EfficientNet, and Xception.

Achieve better performance

- Construct a fine-tuned CNN to classify images from the CIFAR-10 dataset
- Deeper neural network to increase learning capacity, dropout layers, L2 regularization to our convolutional layers, batch normalization layers
- Improve from 66% to 85%.
- See CNN_Cifar10_FineTune.pdf