www.tensorflow.org

1. Libraries :

- import tensorflow as tf
- import tensorflow_datasets as tfds

2. DATASET- Importing

- shuffle_files=True: The MNIST data is only stored in a single file, but for larger datasets with multiple files on disk, it's good practice to shuffle them when training.
- as_supervised=True: Returns a tuple (img, label) instead of a dictionary {'image': img, 'label': label}.

3. DATASET – Normalization

- normalize_img(image, label):
- """Normalizes images: `uint8` -> `float32`."""
- tf.cast - Casts a tensor to a new type.

  tf.cast(x, dtype, name=None): The operation casts x (in case of Tensor) or x.values (in case of SparseTensor or IndexedSlices) to dtype.

4. TRAIN SET – transformations

- tf.data.Dataset.map: TFDS provide images of type tf.uint8, while the model expects tf.float32. Therefore, you need to normalize images.
- tf.data.Dataset.cache As you fit the dataset in memory, cache it before shuffling for a better performance.
  Note: Random transformations should be applied after caching.
- tf.data.Dataset.shuffle: For true randomness, set the shuffle buffer to the full dataset size.
  Note: For large datasets that can't fit in memory, use buffer_size=1000 if your system allows it.
- tf.data.Dataset.batch: Batch elements of the dataset after shuffling to get unique batches at each epoch.
- tf.data.Dataset.prefetch: It is good practice to end the pipeline by prefetching for performance.

5. TEST SET

Testing pipeline is similar to the training pipeline with small differences:

- Don't need to call tf.data.Dataset.shuffle.
- Caching is done after batching because batches can be the same between epochs.

- The model is sequential and not parallel, the same type we learned in DL lecture.

- Module: tf.keras.layers: Keras layers API.
- tf.keras.layers.Flatten(data_format=None, **kwargs)

  class Flatten: Flattens the input. Does not affect the batch size.

data_format	A string, one of channels_last (default) or channels_first. The ordering of the dimensions in the inputs.

- class Dense: Just your regular densely-connected NN layer.

units	Positive integer, dimensionality of the output space.
kernel_initializer	Initializer for the kernel weights matrix
activation	Activation function to use. If you don't specify anything, no activation is applied (ie. "linear" activation: a(x) = x).

- class tf.keras.layers.Dropout: Applies Dropout to the input.

rate	Float between 0 and 1. Fraction of the input units to drop.

- class tf.keras.layers.BatchNormalization: Layer that normalizes its inputs.

  Batch normalization applies a transformation that maintains the mean output close to 0 and the output standard deviation close to 1.

7. ביצוע קומפילציה לאימון

- Class Compile - Configures the model for training.

optimizer	String (name of optimizer) or optimizer instance.
loss	Loss function. May be a string (name of loss function), or a tf.keras.losses.Loss instance.
metrics	List of metrics to be evaluated by the model during training and testing. Each of this can be a string (name of a built-in function), function or a tf.keras.metrics.Metric instance.

- tf.keras.losses.SparseCategoricalCrossentropy:

  Computes the crossentropy loss between the labels and predictions.

- tf.keras.optimizers.Adam:

  Optimizer that implements the Adam algorithm.

learning_rate	A tf.Tensor, floating point value, a schedule that is a tf.keras.optimizers.schedules.LearningRateSchedule, or a callable that takes no arguments and returns the actual value to use. The learning rate. Defaults to 0.001

- tf.keras.metrics.SparseCategoricalAccuracy

  Calculates how often predictions match integer labels.

8. ביצוע אימון למודל

model.fit - trains the model for a fixed number of epochs (iterations on a dataset).

ds_train	Input data.
epochs	Integer. Number of epochs to train the model.
validation_data	Data on which to evaluate the loss and any model metrics at the end of each epoch. The model will not be trained on this data.