Beyond Perceptron: Stepping into the Depths of Learning

Computation Intelligence and its Application in Mechatronics

Winter 2025

Amirkabir University of Technology
(Tehran Polytechnic)

What is an MLP?

- A type of feedforward neural network with multiple layers.
- Consists of:
  - **Input layer**: Receives input features.
  - **Hidden layers**: Extract higher-level patterns.
  - **Output layer**: Produces predictions.
- Uses **activation functions** to introduce non-linearity.

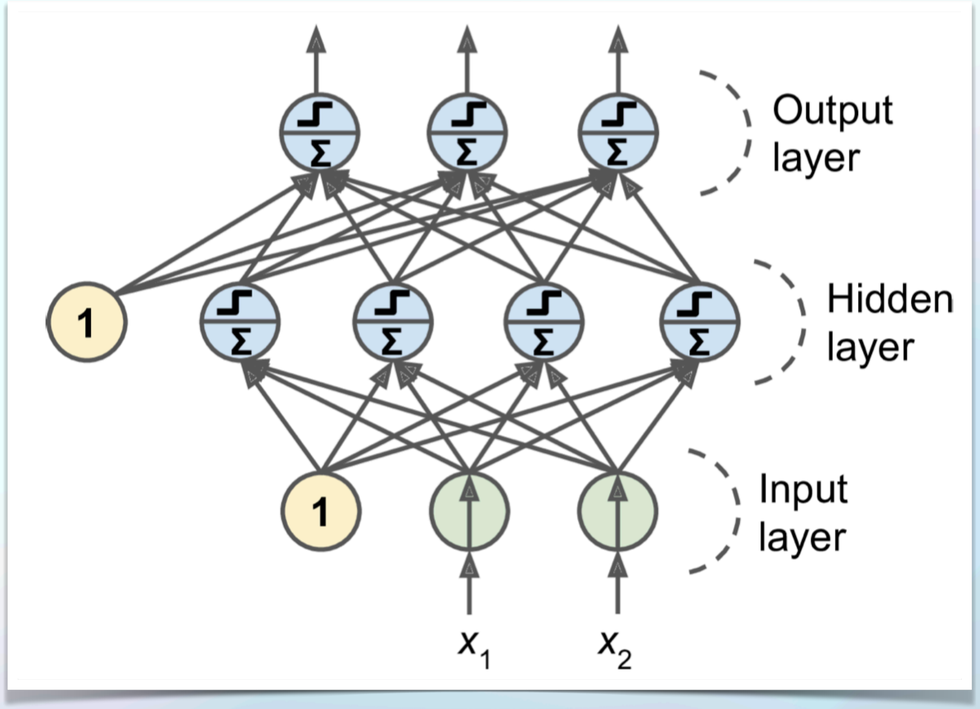


Image from: Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow

Learning Algorithm

Forward and Backpropagation

- **Forward pass:** Computes predictions based on input and weights.
- **Loss function:** Measures error between prediction and target.
- **Backward pass (Backpropagation):**
    - Computes gradients of loss w.r.t weights using **chain rule**.
    - Updates weights using **gradient descent**.

Learning Algorithm

Forward and Backpropagation

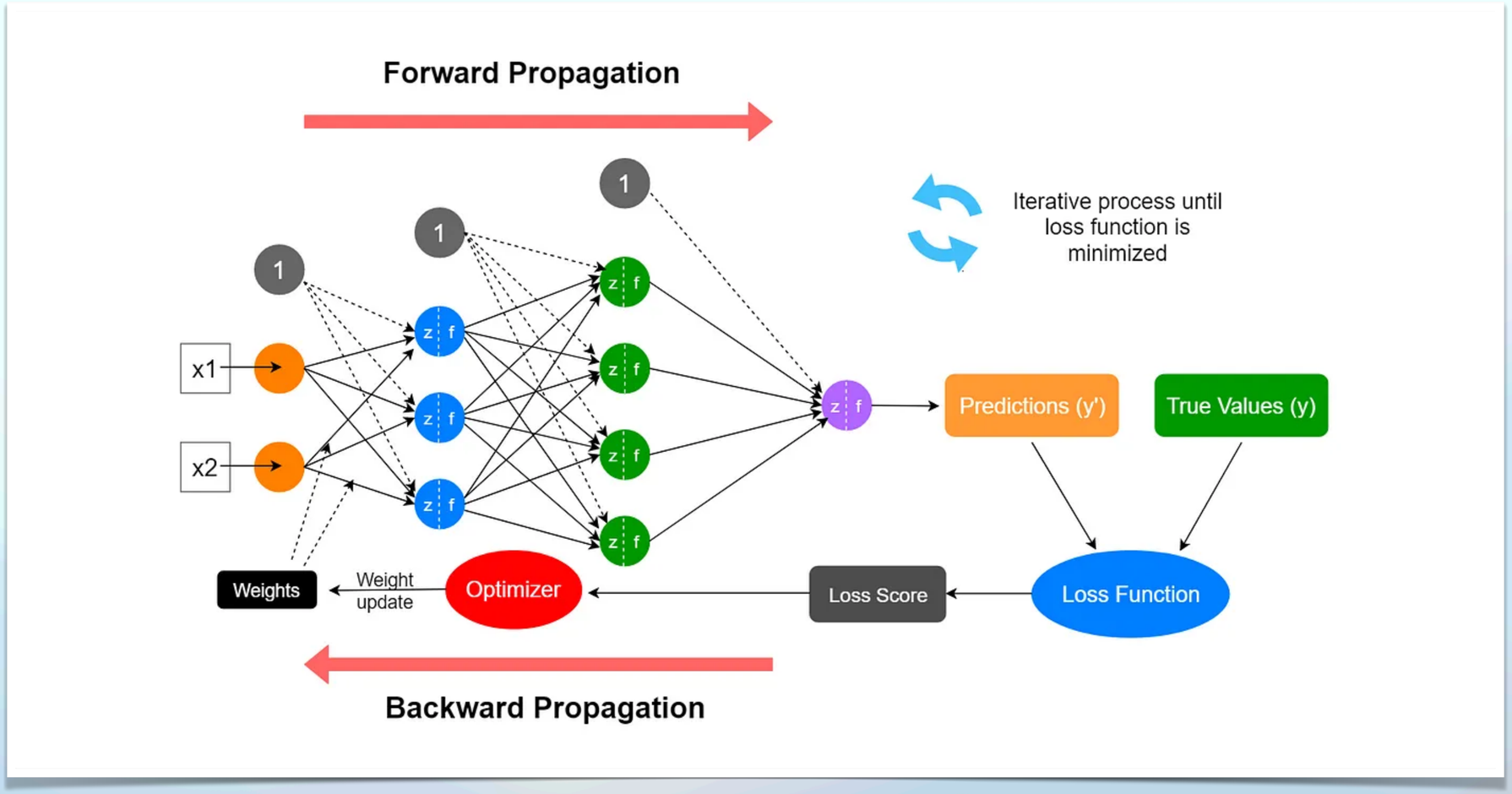
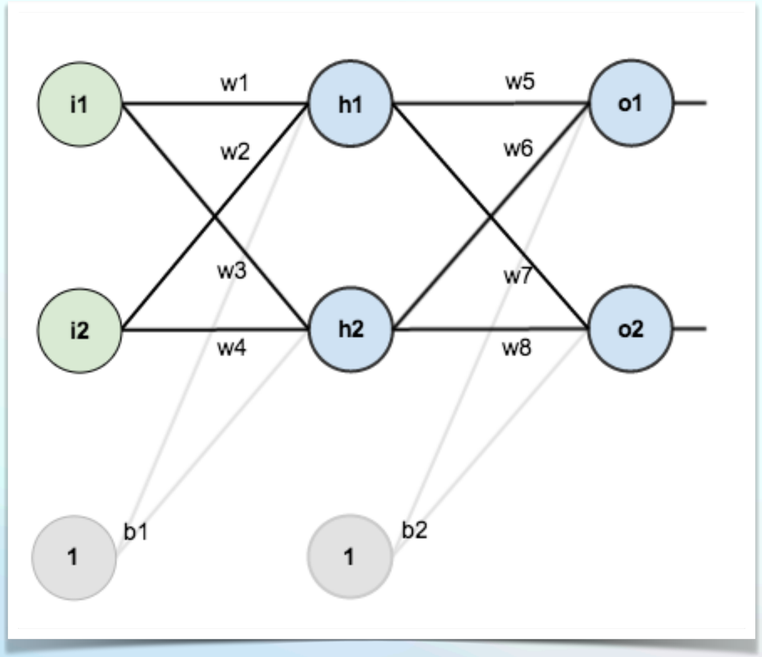


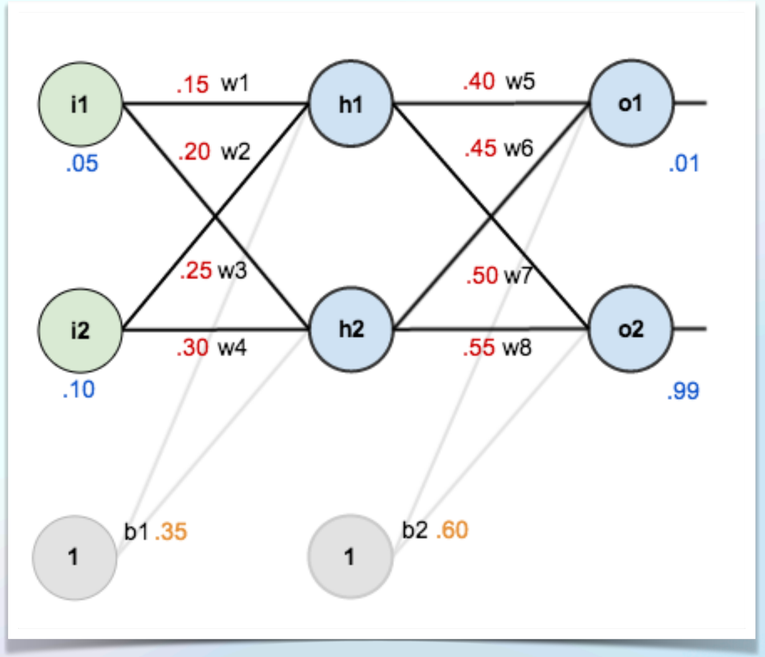
Image from: https://medium.com/@xsankalp13/a-beginners-guide-to-neural-networks-forward-and-backward-propagation-explained-a814666c73ab

Learning Algorithm

A Step by Step Backpropagation Example



Step1: Forward Pass



Reference: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Learning Algorithm

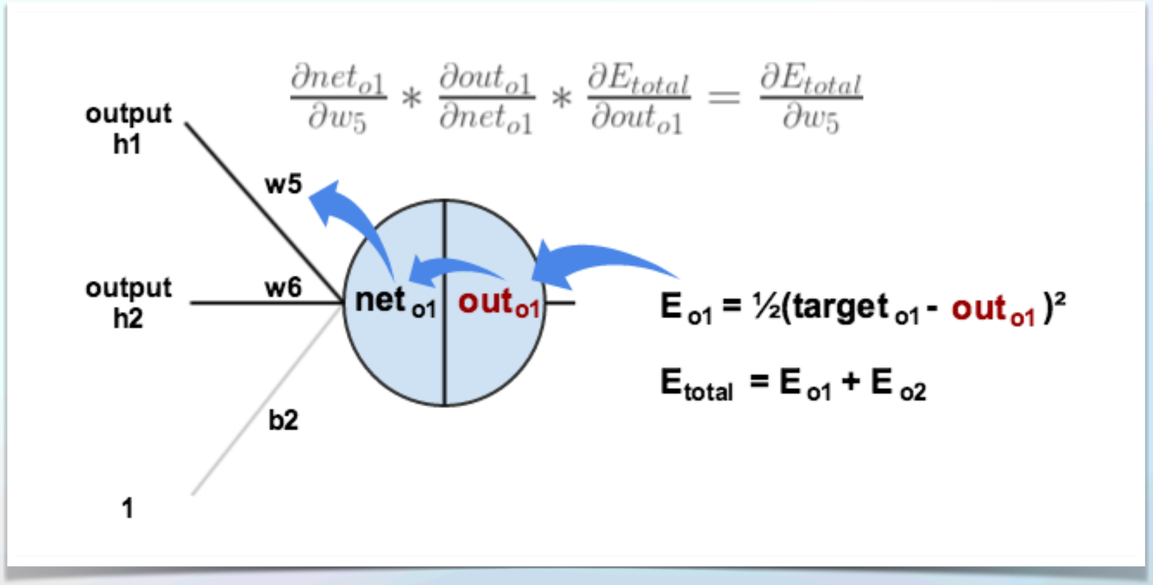
A Step by Step Backpropagation Example

Step 2: Calculating the Total Error → Step 3: The Backwards Pass - Output Layer

$$E_{total} = E_{o1} + E_{o2}$$

$$E_{o1} = \frac{1}{2}(target_{o1} - out_{o1})^2$$

$$E_{o2} = \frac{1}{2}(target_{o2} - out_{o2})^2$$

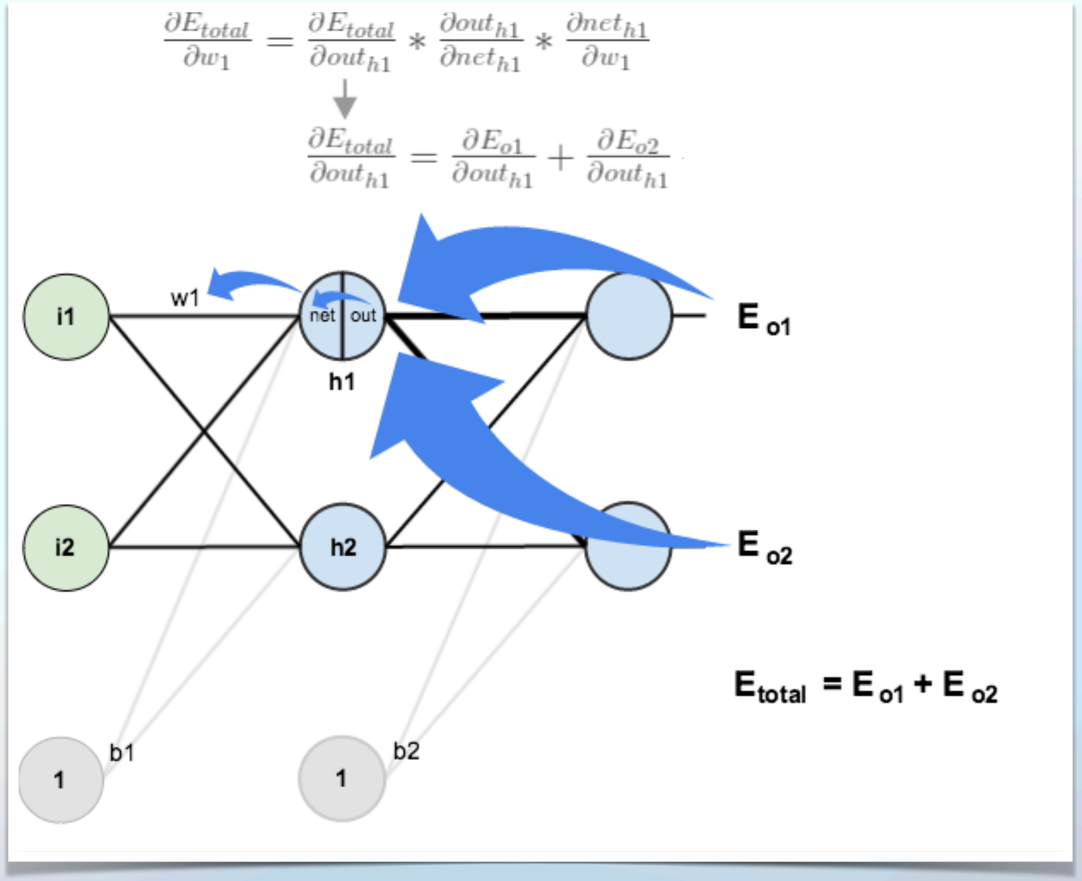


Reference: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Learning Algorithm

A Step by Step Backpropagation Example

Step 4: The Backwards Pass - Hidden Layers



Reference: https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Learning Algorithm

Batch Learning Pseudo-Code

```
Initialize network parameters (weights and biases)
Define loss function and optimizer

For each epoch:
    Shuffle training data
    Divide training data into batches

    For each batch:
        Forward pass: Compute predictions for the batch
        Compute loss using loss function
        Backpropagation: Compute gradients of loss w.r.t. weights
        Apply optimizer to update weights

    Compute validation loss and metrics (if validation data is provided)
    Display training progress (loss, accuracy, etc.)

Return trained model
```

Image from: Vecteezy.com

“The Perceptron’s step activation function was simple and effective for linearly separable problems. But there’s a problem—it’s not differentiable! To enable learning through backpropagation, we need activation functions that are smooth and differentiable. Let’s explore our options.”

Activation Functions in MLP

Basic Activation Functions

Logistic (Sigmoid) Function

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Hyperbolic Tangent (Tanh) Function

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

Rectified Linear Unit (ReLU)

$$\text{ReLU}(x) = \max(0,x)$$

Softmax Function (Multi-Class Classification)

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}}$$

Activation Functions in MLP

Basic Activation Functions

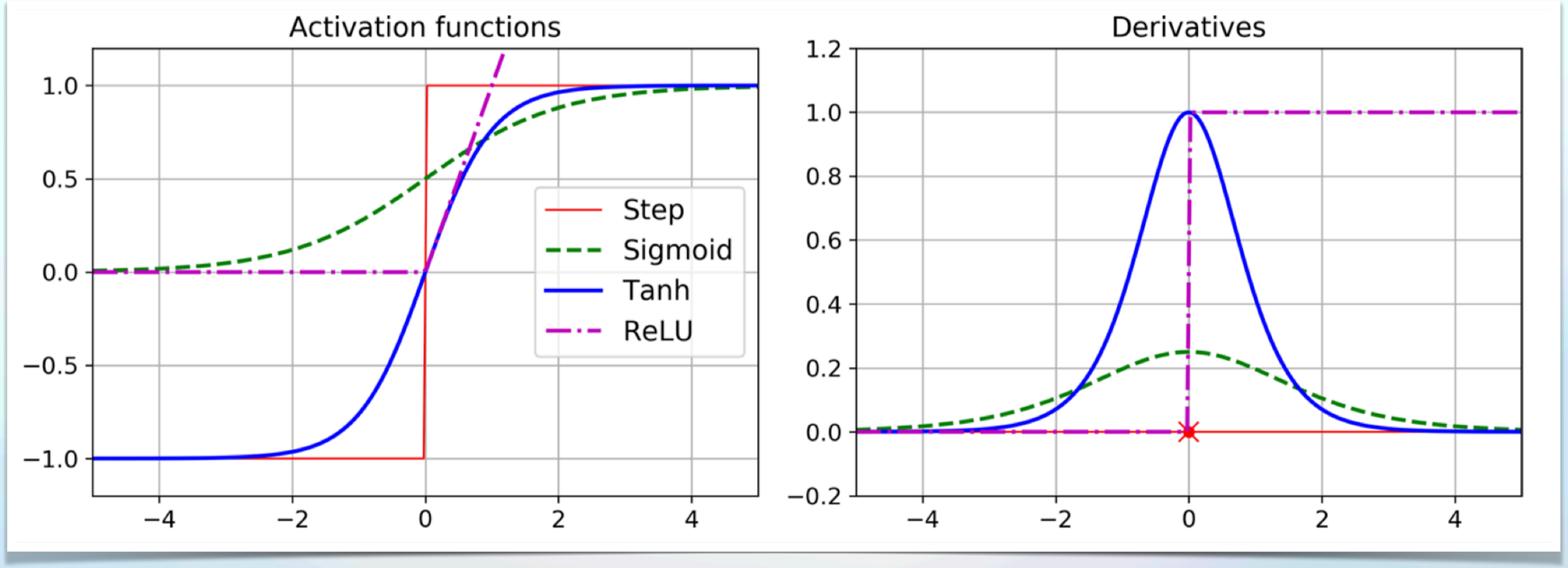


Image from: Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow

Image from: Vecteezy.com

From Theory to Practice: Building MLPs in Keras

Building Your Model

Sequential vs. Functional

The **Sequential model**, which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away).

The **Functional API**, which is an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry strength" model.

Building Your Model

Sequential vs. Functional

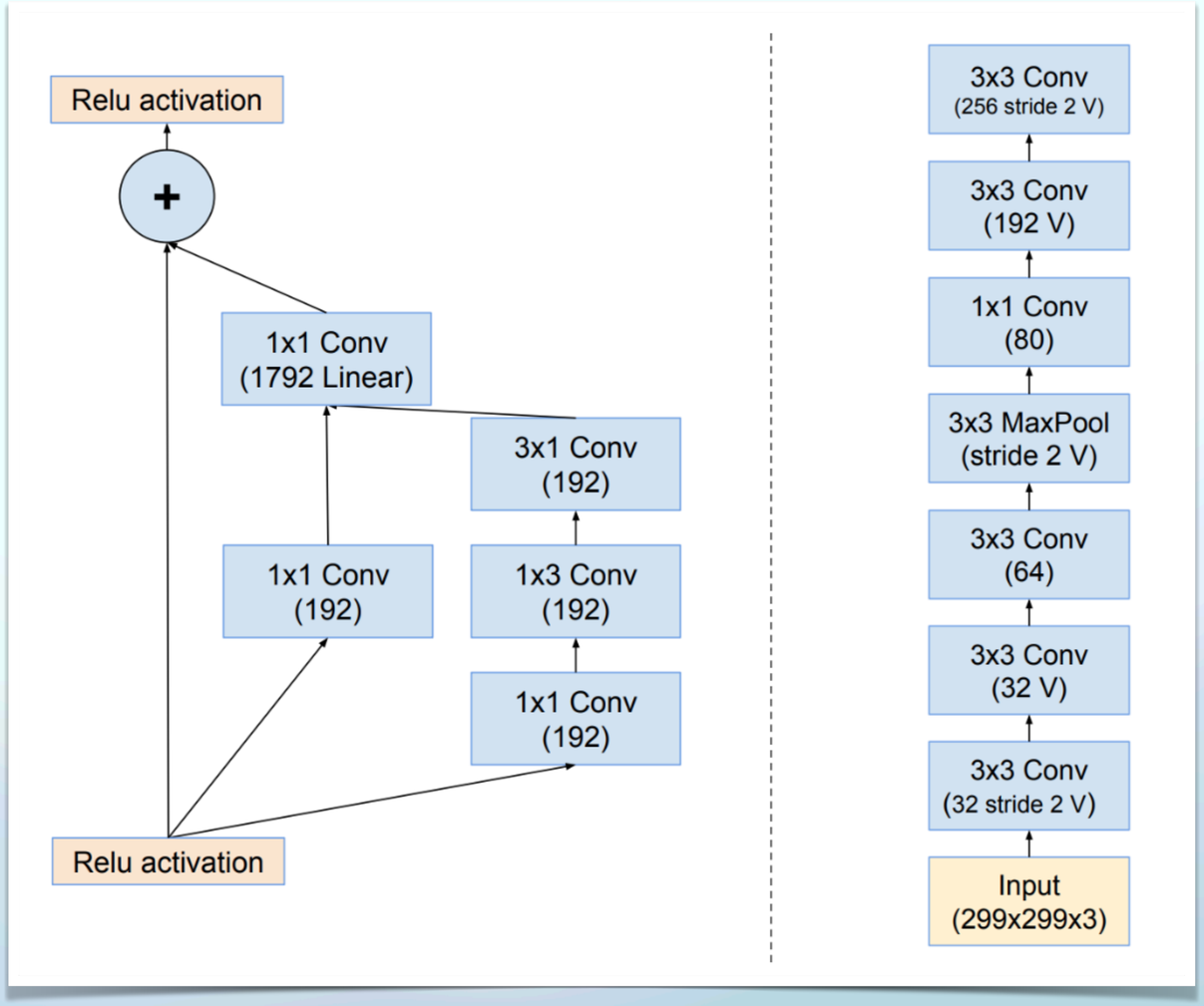


Image from: https://blog.tensorflow.org/2019/01/what-are-symbolic-and-imperative-apis.html

From Building to Predicting

1. Building
2. Compiling
3. Fitting
4. Evaluating
5. Predicting

Can we trust backpropagation to work efficiently in deep networks?

What’s Coming Next

- The Vanishing & Exploding Gradient Problem
- Faster Optimizers: Beyond Basic Gradient Descent
- Avoiding Overfitting: Regularization Techniques