

ML Concepts

Home

Crash Course

Filter

<> Exercises

Help Center

ML models

- Linear regression (70 min)
- Logistic regression (35 min)
- Classification (70 min)

Data

- Working with numerical data (85 min)
- Working with categorical data (50 min)
- Datasets, generalization, and overfitting (105 min)

Advanced ML models

- Neural networks (75 min)
 - Introduction (5 min)
 - Nodes and hidden layers (15 min)
 - Activation functions (10 min)
 - Training using backpropagation (10 min)
- Interactive exercises (15 min)
 - Multi-class classification (10 min)
- Test your knowledge (10 min)
 - What's next
- Embeddings (45 min)
- Large language models (LLMs) (45 min)

Real-world ML



Home > Products > Machine Learning > ML Concepts > Crash Course

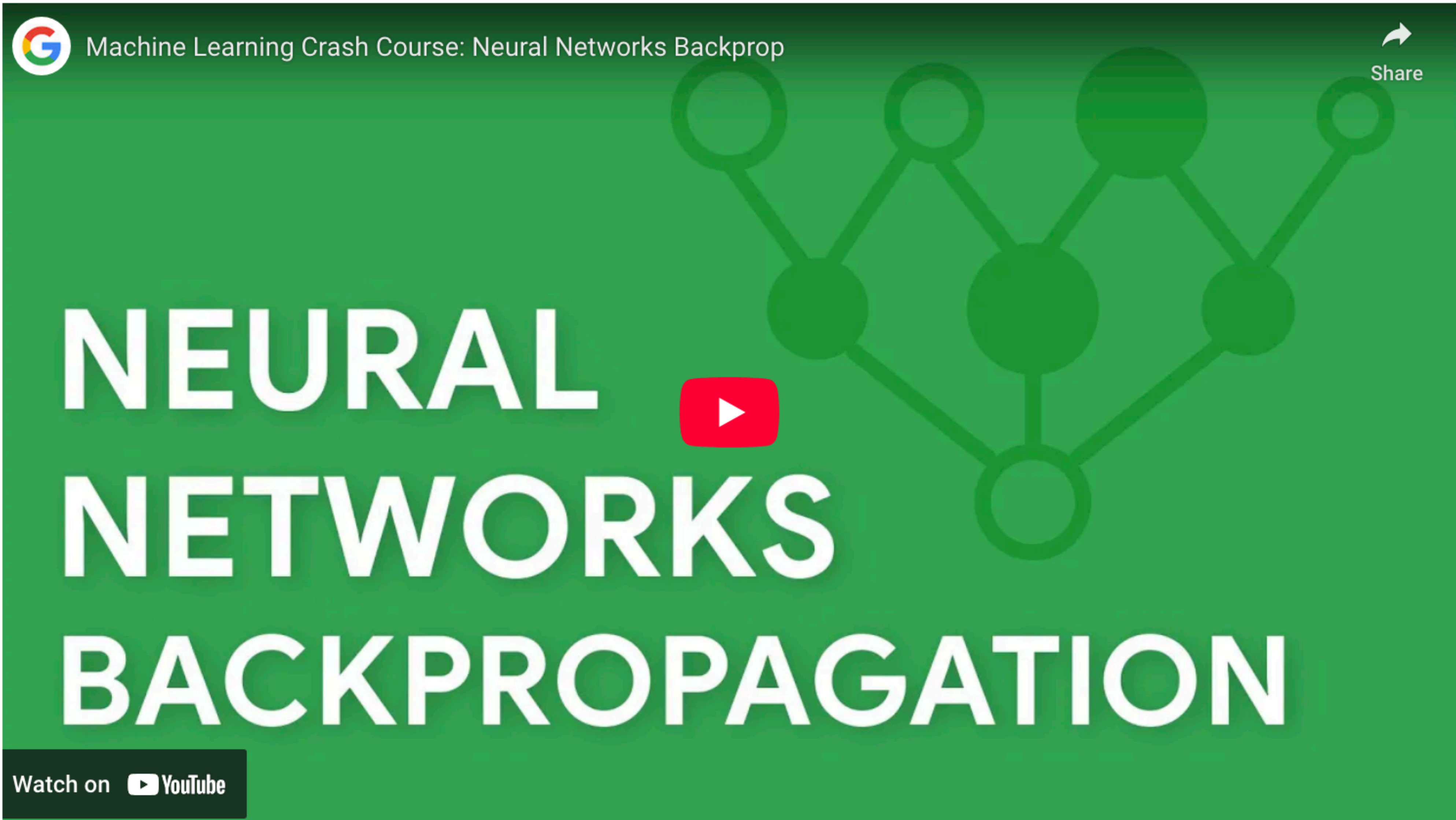
Was this helpful?

Neural Networks: Training using backpropagation



Send feedback

Backpropagation is the most common training algorithm for neural networks. It makes gradient descent feasible for multi-layer neural networks. Many machine learning code libraries (such as [Keras](#)) handle backpropagation automatically, so you don't need to perform any of the underlying calculations yourself. Check out the following video for a conceptual overview of how backpropagation works:



★ To learn more about building image models, check out the [Image Classification](#) course.

Best practices for neural network training

This section explains backpropagation's failure cases and the most common way to regularize a neural network.

★ **NOTE:** The backpropagation training algorithm makes use of the calculus concept of a [gradient](#) to adjust model weights to minimize loss. Understanding and debugging the issues below usually requires some background in calculus.

Vanishing Gradients

The **gradients** for the lower neural network layers (those closer to the input layer) can become very small. In **deep networks** (networks with more than one hidden layer), computing these gradients can involve taking the product of many small terms.

When the gradient values approach 0 for the lower layers, the gradients are said to "vanish". Layers with vanishing gradients train very slowly, or not at all.

The ReLU activation function can help prevent vanishing gradients.

Exploding Gradients

If the weights in a network are very large, then the gradients for the lower layers involve products of many large terms. In this case you can have exploding gradients: gradients that get too large to converge.

Batch normalization can help prevent exploding gradients, as can lowering the learning rate.

Dead ReLU Units

Once the weighted sum for a ReLU unit falls below 0, the ReLU unit can get stuck. It outputs 0, contributing nothing to the network's output, and gradients can no longer flow through it during backpropagation. With a source of gradients cut off, the input to the ReLU may not ever change enough to bring the weighted sum back above 0.

Lowering the learning rate can help keep ReLU units from dying.

★ There are also many variants of ReLU that were designed to address this specific problem, such as [LeakyReLU](#), which you may want to consider using as an activation function to prevent dead ReLU units.

Dropout Regularization

Yet another form of regularization, called **dropout regularization**, is useful for neural networks. It works by randomly "dropping out" unit activations in a network for a single gradient step. The more you drop out, the stronger the regularization:

- 0.0 = No dropout regularization.
- 1.0 = Drop out all nodes. The model learns nothing.
- Values between 0.0 and 1.0 = More useful.

Key terms:

- [Backpropagation](#)
- [Dropout regularization](#)
- [Exploding gradient problem](#)
- [Vanishing gradient problem](#)

Help Center

Previous

← [Activation functions \(10 min\)](#)

Next

[Interactive exercises \(15 min\)](#) →

Was this helpful?



Send feedback

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](#), and code samples are licensed under the [Apache 2.0 License](#). For details, see the [Google Developers Site Policies](#). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2024-11-08 UTC.

Connect

Blog

Instagram

LinkedIn

X (Twitter)

YouTube

Programs

Google Developer Groups

Google Developer Experts

Accelerators

Women Techmakers

Google Cloud & NVIDIA

Developer consoles

Google API Console

Google Cloud Platform Console

Google Play Console

Firebase Console

Actions on Google Console

Cast SDK Developer Console

Chrome Web Store Dashboard

Google Home Developer Console

Google for Developers

Android

Chrome

Firebase

Google Cloud Platform

Google AI

All products

Terms | Privacy

Sign up for the Google for Developers newsletter

Subscribe

English ▾