BST 261: Data Science II
Lecture 3

Feedforward networks in Python with Keras, Regularization

Heather Mattie
Harvard T.H. Chan School of Public Health

Spring 2 2020

Recipe of the day

Black Forest Cake

Fancy Black Forest Cake

Paper Presentations

ImageNet: A Large-Scale Hierarchical Image Database (2009)

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li and Li Fei-Fei Dept. of Computer Science, Princeton University, USA

Presented by Daniel Waranch

Importance

- Image classification is a foundational tool for deep learning
- This paper presents a few cool techniques that are easy to understand but very powerful

Ideas of the paper

- Nouns can be classified by synonyms, which we call a synset. For example, person and human can be thought of as one synset (one category for classification.)
- If we had 500-1000 high resolution, clean images for each synset, in theory we could fully determine all nouns in the English language via images.
- This information could provide powerful training and benchmarking data for researchers and algorithms.
- ImageNet seeks to create this database of images

Structure of ImageNet

- ImageNet is named after WordNet which has 80,000 sysnets that ImageNet hopes to populate. At the time of the paper they had 5247 sysnets consisting of 3.2 million images. Today it’s 14 million images documenting 21841 sysnets.
- Hierarchical, “Is-A” relationships. “tree based” structure.

$$\begin{tikzcd}
\boxed{\text{mammal}} \arrow[r] & \boxed{\text{placental}} \arrow[r] & \boxed{\text{carnivore}} \arrow[r] & \boxed{\text{canine}} \arrow[r] & \boxed{\text{dog}} \arrow[r] & \boxed{\text{Working dog}} \arrow[r] & \boxed{\text{husky}}
\end{tikzcd}$$

- Candidate images are found via Internet search (10% accuracy) and then labelled by humans with Mechanical Turk. Generally speaking, the lower on the hierarchy tree, human accuracy is lower (dog. Vs. husky). Statistical methods are used to achieve high level of confidence for human labelling, leading to 99.7% precision.

Comparison with other image databases

- The authors compare this database to other images, and claim that ImageNet is superior due to the following properties:
- Disambiguated Labels: (can tell the difference between a river bank and a financial bank. Largely due to tree structure.)
- Clean Annotations
- Dense Hierarchies (much more subtrees of categories than other databases.)
- Full res. Images
- Publicly available
- ImageNet does not have segmented labelling (can label a tree and a dog in a single image.) There are databases that do this, but they are much smaller.

3 Applications/tests for ImageNet

- Tree based classification: This technique serves to show the importance of the tree based structure of the database.
- Demonstrative example.

(1.) I ask the algorithm "Is this picture a mammal?" I'm 60% sure it is a Mammal.

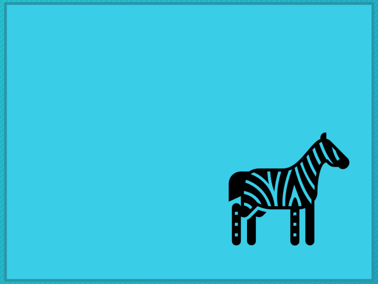
(2.) Ask the algorithm "Is the **same** picture a dog?" I am 80% sure it is a dog.

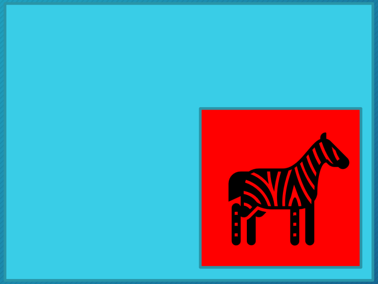
(3.) Because this is a hierarchical tree database, we actually know that true answer to (1.) is actually 80%.

- Because the algorithm is dense and tree based, we can improve our classification quite easily via checking all the sub-nodes of the tree. If a sub-node provides better performance than we can just use that instead!

3 Applications/tests for ImageNet

- Object Localization: Technique to get a more “obvious” image. For example, consider a 1000x1000 pixel image that has a zebra in it. Perhaps the algorithm give it a 95% chance of having a zebra. We can consider many possible “sub-rectangles” of the 1000x1000 image. We feed the sub-rectangles into the algorithm and find the one that gives the highest percent chance to be a zebra, lets say 99%. This “localizes” onto the zebra.





3 Applications/tests for ImageNet

- Object recognition: This is the most “obvious” application. The authors perform 4 tests. 2 with an algorithm with noisy vs. clean data to show the advantage of clean data. And 2 with 100 images per category of training vs. the full category for training to show the importance of lots of images. Results are as we would expect, with the clean data performing better than noisy data, and the full category performing better than the partial category. Furthermore they extract features from the high resolution data to show that high res data performs better than low res data. Performance is plotted by ROC curves.

Conclusion

- ImageNet is a great resource from training and testing image classification algorithms.
- It has many properties that make it useful.
- There really aren’t ***that*** many nouns in the world, so classifying them with images feels like an approachable task.
- We have seen two cool methods that is allowed with the structure of ImageNet.

ImageNet Classification with Deep Convolutional Neural Networks

AUTHORS: ALEX KRIZHEVSKY, ILYA SUTSKEVER, GEOFFREY E. HINTON (2012)

PRESENTATION: GENEVIEVE LYONS

Goal: To improve performance of object-recognition ML (address common barriers)

- Collect larger datasets
- More powerful models
- Preventing overfitting
- Execution time

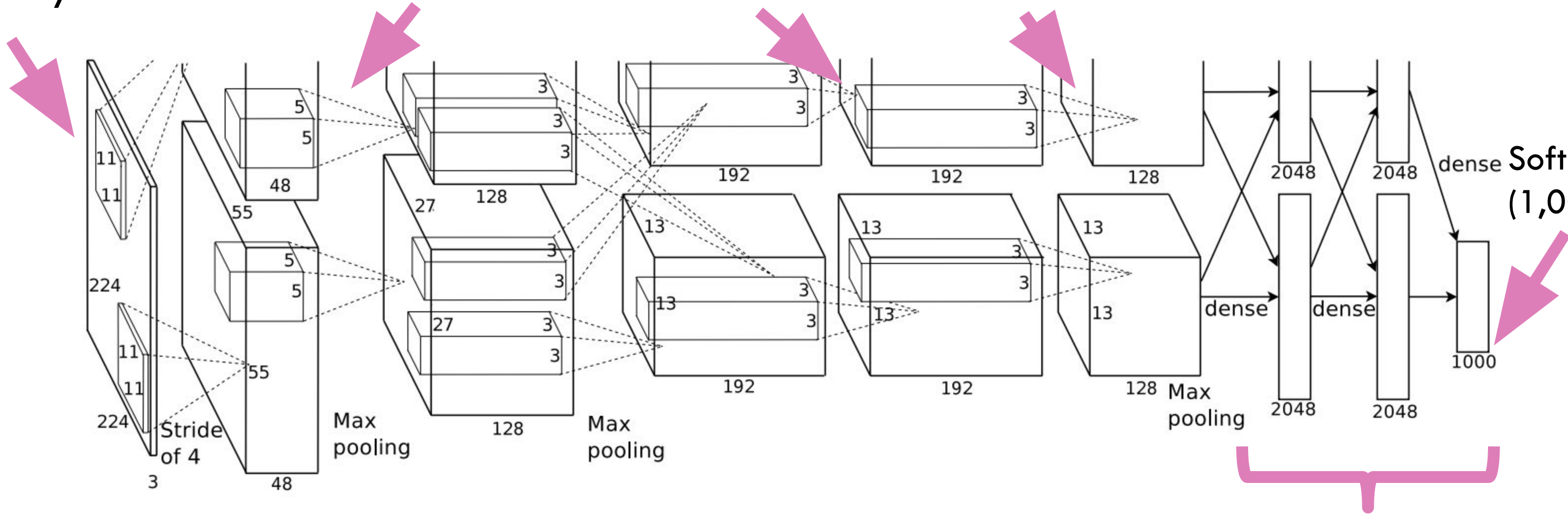
Goal: To improve performance of object-recognition ML

- Collect larger datasets
  - ImageNet: > 15M labeled high-resolution images in > 22,000 categories
- More powerful models
  - CNNs: Appropriate for images (locality of pixels), fewer connections, fewer parameters
- Preventing overfitting
  - Augmenting Data, Dropout layers
- Execution time
  - Highly optimized GPU implementation of 2D convolution and other operations (across 2 GPUs)

Results: One of the largest CNNs to date on ImageNet, with the best results ever reported on these datasets (**AlexNet**)

Input layer

Kernels of 2nd, 4th, and 5th layers only connected to same GPU



Softmax (1,000 classes

First 5 layers are convolutional

Last 3 layers are fully connected

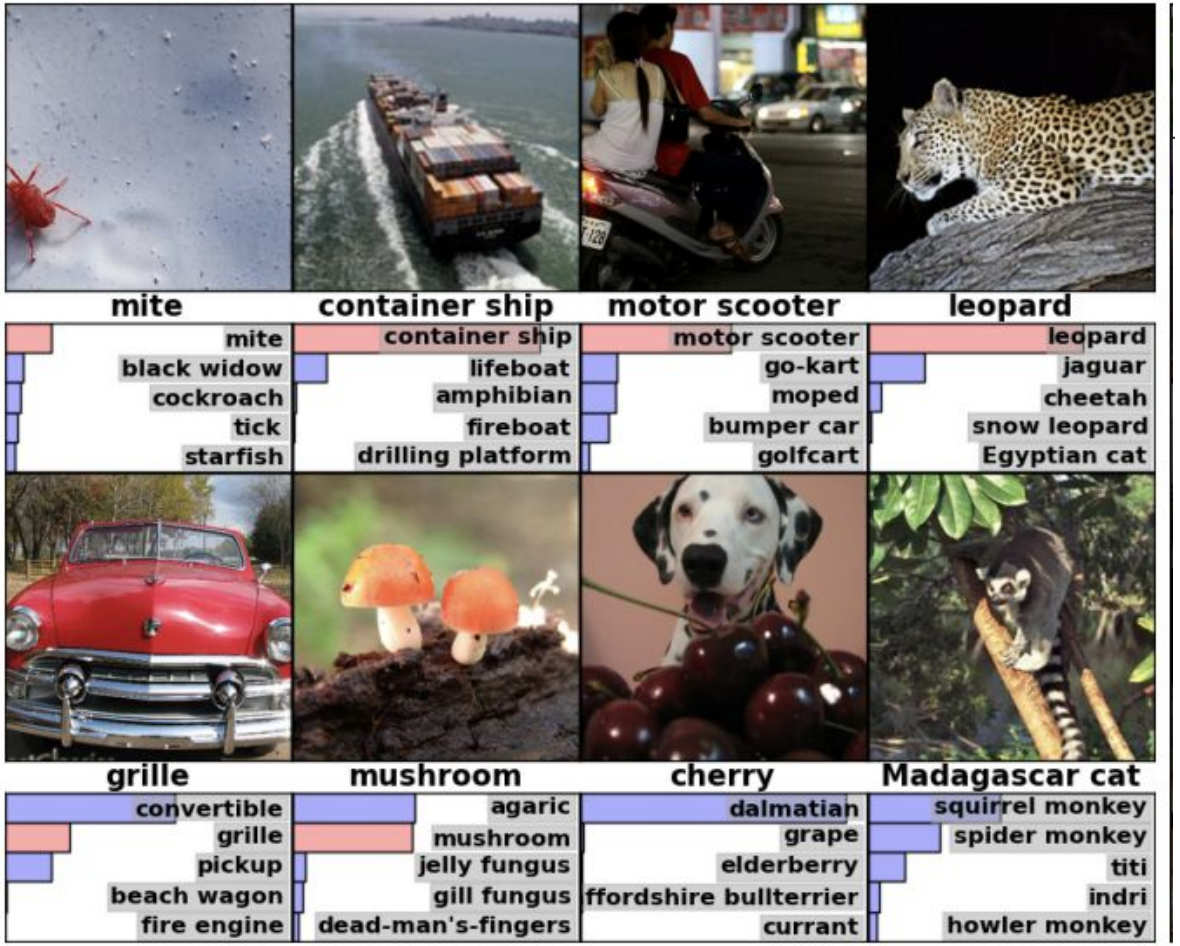
60 million parameters!

Reducing Overfitting:

- Data Augmentation
  - Train network on randomly selected patches of images (increases training data size!)
  - Alter intensity of RGB channels through PCA
- Dropout
  - Randomly exclude neurons ($p = 0.5$) for each new input

Results

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

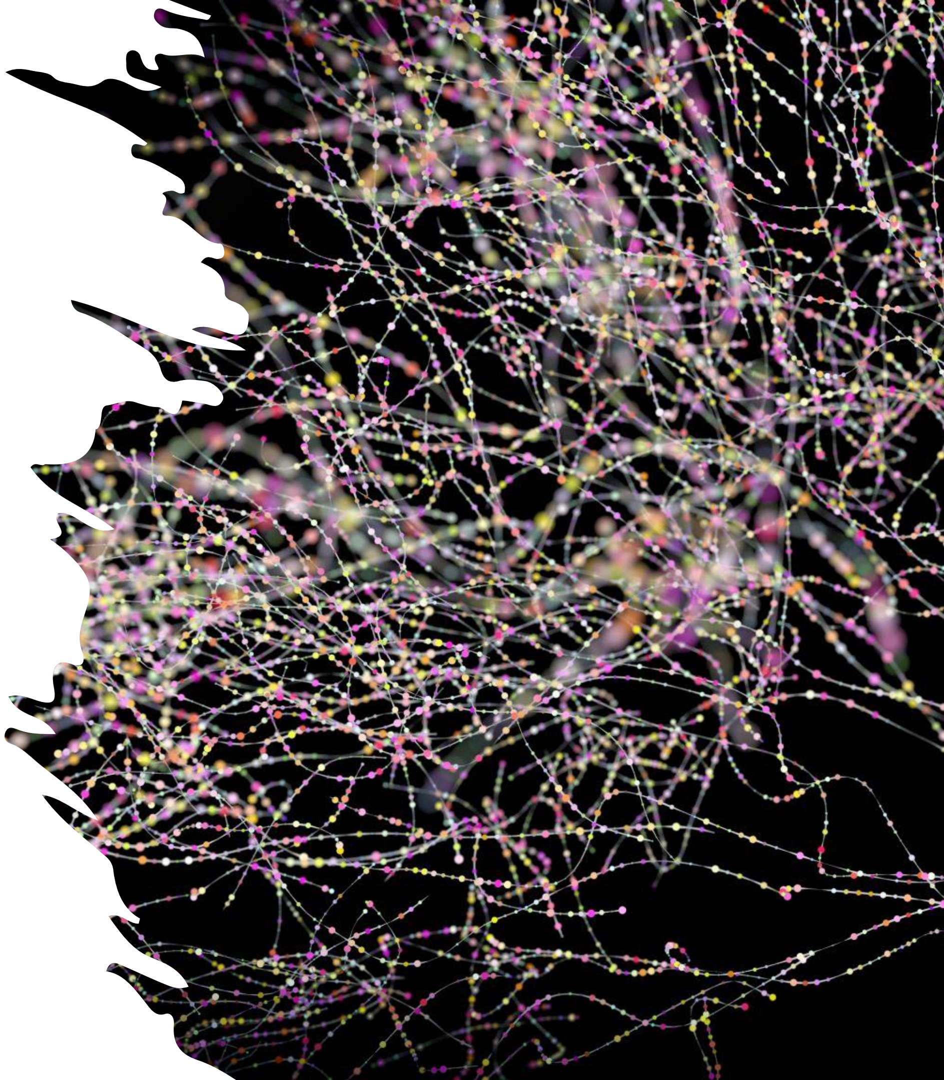


Going Deeper with Convolutions

Szegedy et al.

PRESENTATION BY:

REBECCA YOUNGERMAN

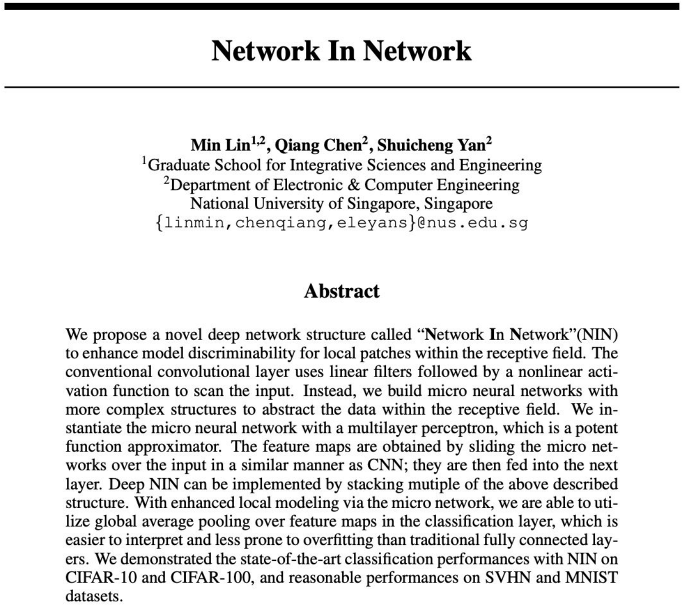


Inception – A New Deep CNN

- Classification and Detection
    - ILSVRC 2014

- Width and Depth
- Computational Budget Constant

- Sparse Substructure



Network In Network

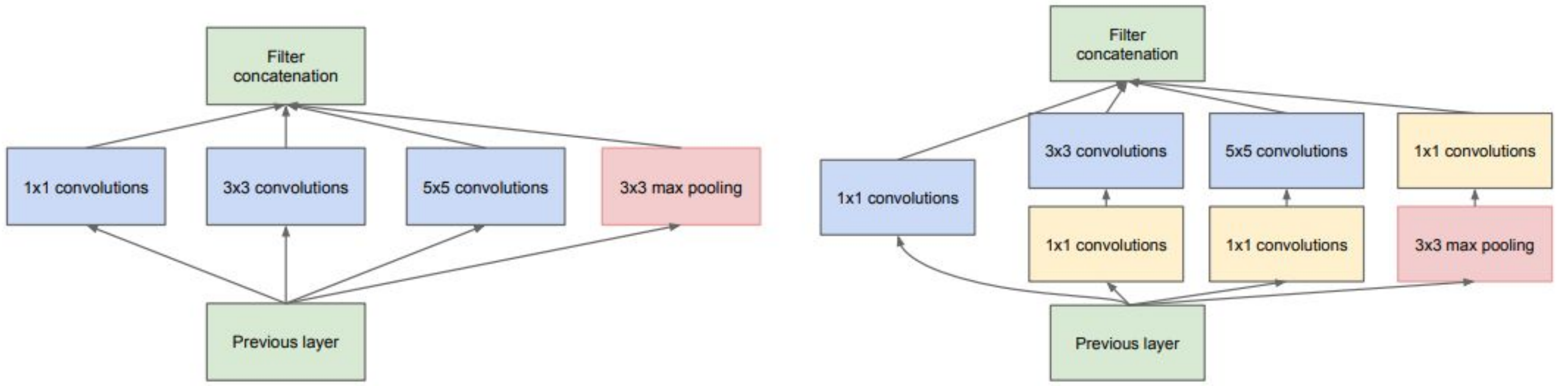
Min Lin[1,2], Qiang Chen[2], Shuicheng Yan[2]
[1]Graduate School for Integrative Sciences and Engineering
[2]Department of Electronic & Computer Engineering
National University of Singapore, Singapore
{linmin,chenqiang,eleyans}@nus.edu.sg

Abstract

We propose a novel deep network structure called "**N**etwork **I**n **N**etwork"(NIN) to enhance model discriminability for local patches within the receptive field. The conventional convolutional layer uses linear filters followed by a nonlinear activation function to scan the input. Instead, we build micro neural networks with more complex structures to abstract the data within the receptive field. We instantiate the micro neural network with a multilayer perceptron, which is a potent function approximator. The feature maps are obtained by sliding the micro networks over the input in a similar manner as CNN; they are then fed into the next layer. Deep NIN can be implemented by stacking mutiple of the above described structure. With enhanced local modeling via the micro network, we are able to utilize global average pooling over feature maps in the classification layer, which is easier to interpret and less prone to overfitting than traditional fully connected layers. We demonstrated the state-of-the-art classification performances with NIN on CIFAR-10 and CIFAR-100, and reasonable performances on SVHN and MNIST datasets.

+

Save the computation – dimension reduction



(a) Inception module, naïve version (b) Inception module with dimension reductions

Figure 2: Inception module

GoogLeNet – Better than everyone else's LèNet

- ReLu!
- Computational efficiency and practicality
- Auxiliary networks for backpropagation concerns

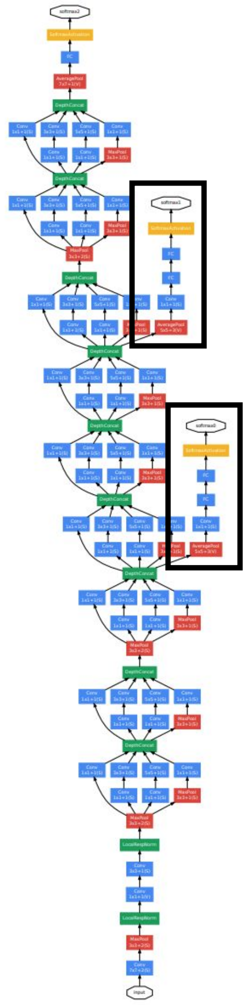


Figure 3: GoogLeNet network with all the bells and whistles

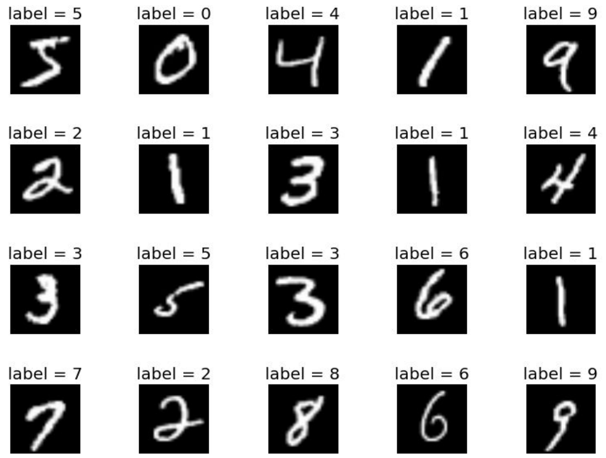
References

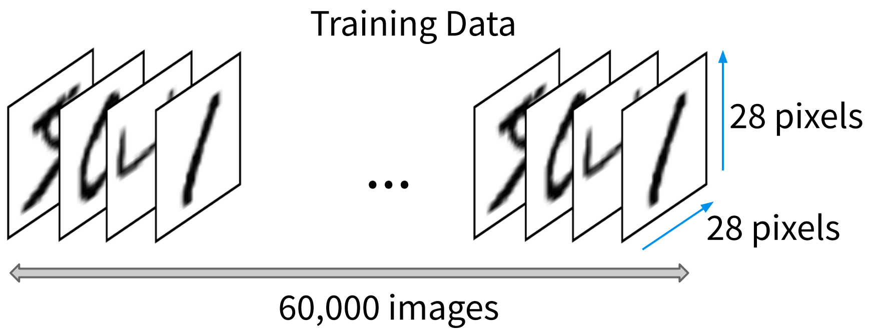
- Going Deeper with Convolutions: https://arxiv.org/pdf/1409.4842.pdf
- Inception Meme: http://knowyourmeme.com/memes/we-need-to-go-deeper

MLPs in Python/Keras

MNIST Data Example

- The MNIST data set includes handwritten digits with corresponding labels
- Training set: 60,000 images of handwritten digits and corresponding labels
  - Each digit is represented as a 28 x 28 matrix of grayscale values 0 - 255
  - The entire training set is stored in a 3D tensor of shape (60000, 28, 28)
  - The corresponding image values are stored as a 1D tensor of values 0 - 9
- Testing set: 10,000 images with the same set up as the training set





MNIST Data Example

Data wrangling

- We’ll get into RGB images later, but for grayscale images, we need to first transform the matrix of values into a vector of values, and then normalize them to be between 0 and 1. It is not strictly necessary to normalize your inputs, but smaller numbers help speed up training and avoid getting stuck in local minima. This also ensures the gradients don’t “explode” or “vanish”
  - Reshape each image from a 28 x 28 matrix of grayscale values 0 - 255 to a vector of length 28*28 = 784 of values 0 - 1 (divide each by 255)

- We now have 10 classes (categories; the digits 0-9)
  - We need to have multiclass labels that tell the network which digit the example is
  - Reshape each corresponding image label to a vector of length 10 of values 0 or 1
  - Example: the digit 3 would be represented as [0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
  - You can think of this as “dummy coding” the labels

Activation and Loss Function Choices

Task	Last-layer activation	Loss function
Binary classification	sigmoid	Binary cross-entropy
Multiclass, single-label classification	softmax	Categorical cross-entropy
Multiclass, multilabel classification	sigmoid	Binary cross-entropy
Regression to arbitrary values	None	Mean square error (MSE)
Regression to values between 0 and 1	sigmoid	MSE or binary cross-entropy

Softmax function

$$\text{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

- Softmax units are used as outputs when predicting a discrete variable *y* with *j* possible values
- In this setting, which can be seen as a generalization of the Bernoulli distribution, we need to produce a vector $\hat{\mathbf{y}}$ with $\hat{y}_i = P(y = i|x)$
- We require that each $\hat{y}_i$ lie in the [0, 1] interval and that the entire vector sums to 1
- We first compute $z = w^T x + b$ as usual
- Here, $z_i = log[\tilde{P}(y = i|x)]$ represents an unnormalized log probability for class *i*
- The softmax function then exponentiates and normalizes *z* to obtain $\hat{\mathbf{y}}$

Categorical cross-entropy

- In this case we want to maximize

$$log[P(y=i;z)] = log[\text{softmax}(z)_i] = z_i - log\sum_j exp(z_j)$$

- The first term shows that the input always has a direct contribution to the loss function

- Because $log\sum_j exp(z_j) \approx max_j z_j$, the negative log-likelihood loss function always strongly penalizes the most active incorrect prediction

MNIST Data Example

Network Architecture

- Let’s start with 2 layers:
  - Hidden layer will have 512 hidden units and the **relu activation function**

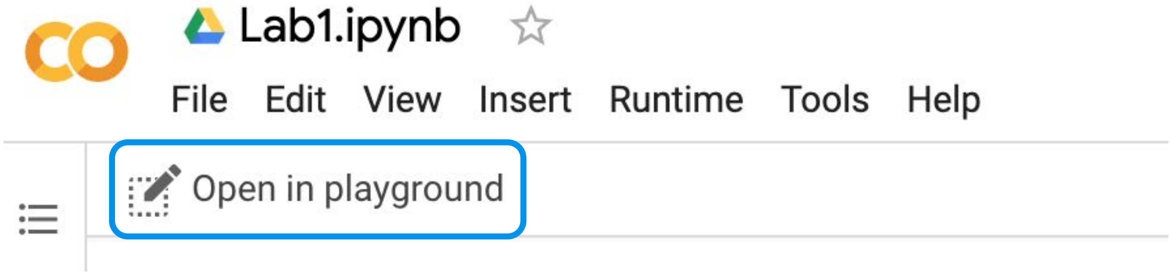
  - Output layer with 10 units (one for each possible digit) and the **softmax activation function** (this produces a vector of length 10, where each element is a probability between 0 and 1 of the image being classified as that digit)
  - Example: [0, 0.3, 0, 0, 0, 0, 0, 0.7, 0, 0] - the highest probability corresponds to a label of 7, so the network would classify this image as a 7

  - **rmsprop optimization algorithm**
  - **categorical_crossentropy loss function**
  - **accuracy performance measure** (the proportion of times the correct class is chosen)

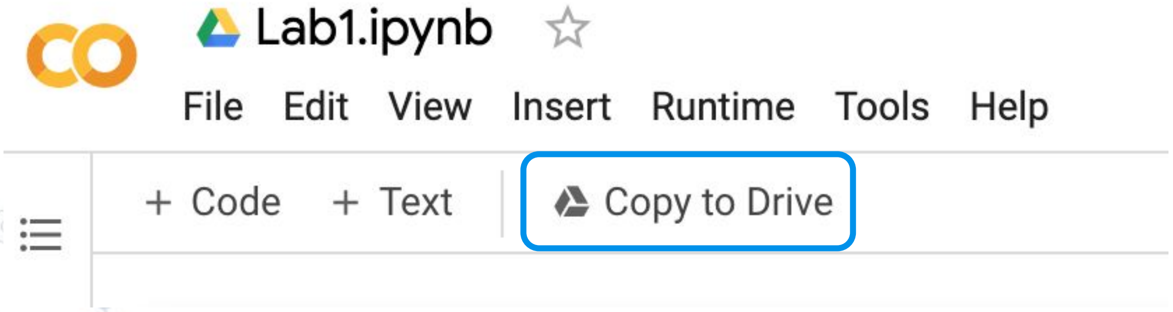
MNIST Data Example

Colab link

Step 1



Step 2


Lab1.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
Copy to Drive