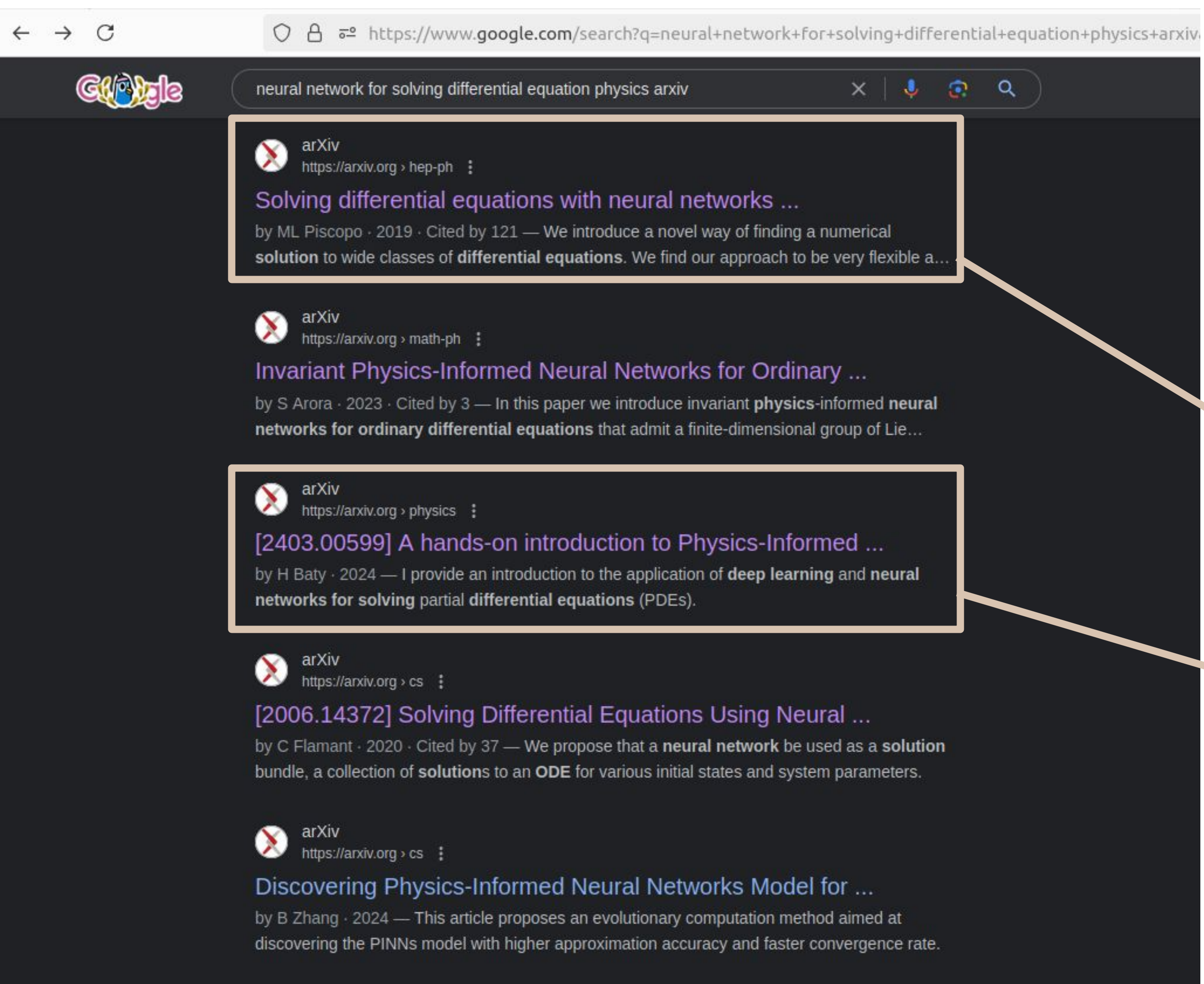
PHY654

Machine learning (ML) in particle physics

Swagata Mukherjee • IIT Kanpur
5th and 7th September 2024

Solving DE using NN: an active research area



PINN: Physics-Informed Neural Networks

Applications to the calculation of cosmological phase transitions. *Phys. Rev. D 100, 016002 (2019)*

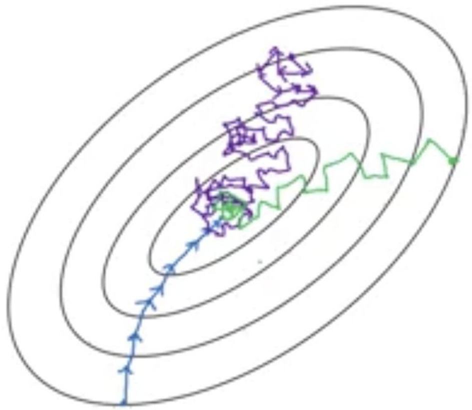
Laplace equation,
Poisson equations,
Helmholtz equations,
Grad-Shafranov equations,
etc

Mini-batch gradient descent

What if m is VERY large? Eg: 5 or 10 Million.
m=number of training example.

In that case, it's difficult to process all m examples together, instead do mini-batch gradient descent.

Split your large training set into several mini training sets (mini-batches).



Mini-batch size: what to choose?
Two extremes
Mini-batch size: m → Batch Gradient Descent
Mini-batch size: 1 → Stochastic Gradient Descent (very noisy)
In practise we use something in between.

Generally one can try mini-batch sizes of 64 or 128 or ….. 1024

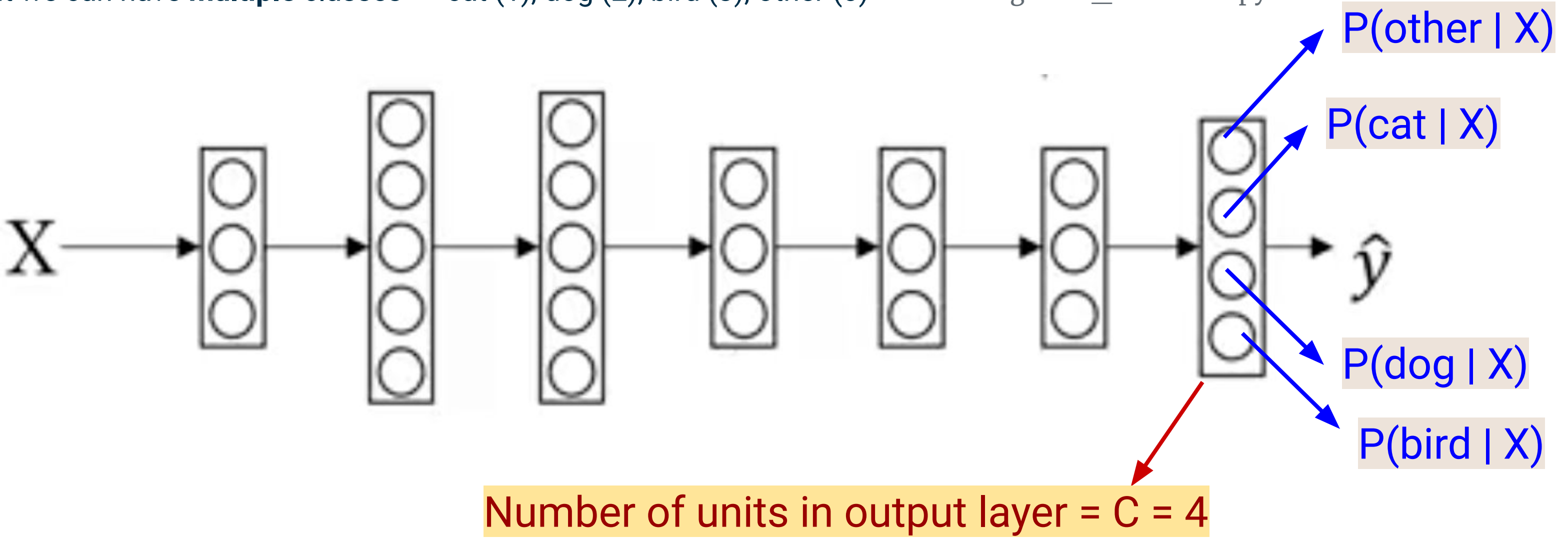
The way computer memory is laid out and accessed, code may run faster if mini-batch size is 2^x

Multi-class classification

So far we discussed **binary** classification → cat (1) vs non-cat (0)
But we can have **multiple** classes → cat (1), dog (2), bird (3), other (0)

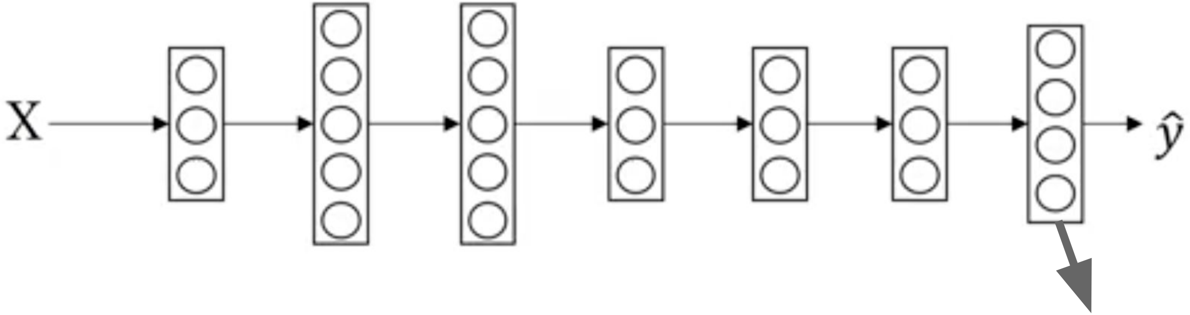
loss='binary_crossentropy'
loss='categorical_crossentropy'

C = number of classes = 4



ŷ is (4,1) dimensional vector

Softmax activation function



$$z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \qquad t = \begin{bmatrix} e^5 \\ e^2 \\ e^{-1} \\ e^3 \end{bmatrix}$$

$$\mathbf{a^{[l]}} = \; g^{[L]}(z^{[L]}) = \begin{bmatrix} e^5/(e^5 + e^2 + e^{-1} + e^3) \\ e^2/(e^5 + e^2 + e^{-1} + e^3) \\ e^{-1}/(e^5 + e^2 + e^{-1} + e^3) \\ e^3/(e^5 + e^2 + e^{-1} + e^3) \end{bmatrix} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

Non-numerical data

```
[1] from sklearn.preprocessing import LabelEncoder
```

```
colors = ['red', 'blue', 'green', 'red', 'green']
encoder = LabelEncoder()

encoded_colors = encoder.fit_transform(colors)
print(encoded_colors)
```

```
[2 0 1 2 1]
```

Use label encoding to convert non-numerical data into numerical form.

One-hot encoding

```
[10] import numpy as np
     y_real = np.array([0, 0, 1, 2, 2, 1, 1, 0, 1, 2])
     type(y_real)
```

```
numpy.ndarray
```

```
[17] print (y_real)
```

```
[0 0 1 2 2 1 1 0 1 2]
```

```
encoder_y = OneHotEncoder(sparse_output=False)
y_real_oh = encoder_y.fit_transform(y_real.reshape(-1,1))
```

```
print (y_real_oh)
```

```
[[1. 0. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]
 [0. 0. 1.]
 [0. 1. 0.]
 [0. 1. 0.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Use one hot encoding to convert categorical or numerical variables into binary vectors.

Feature normalization

```
[35] X =            ([200.9, 0.04],
                    [205.1, 0.01],
                    [223.2, 0.09],
                    [254.0, 0.10]
                    )

     from sklearn.preprocessing import StandardScaler
     sc = StandardScaler()
     X_normalized = sc.fit_transform(X)
```

```
print (X_normalized)
```

```
[[-0.95126609 -0.54433105]
 [-0.75049636 -1.36082763]
 [ 0.11472556  0.81649658]
 [ 1.58703689  1.08866211]]
```

Use standard scaler to normalize features (X).

Tensorflow and Keras

- TensorFlow is an open-source library for ML.
    - Offers high level of flexibility
    - You can define every aspect of your NN architecture and training process.
    - But flexibility comes at a cost. Sometimes, it can be challenging for beginners to grasp
- Keras is an open-source library for ML that runs on top of Tensorflow.
    - Keras is designed to be user-friendly
    - Excellent choice for newcomers to deep learning.

Keras: example code

XOR Gate
https://github.com/swagata87/IITKanpurPhy654/blob/main/XOR_NN_keras.ipynb

Ising model
https://github.com/swagata87/IITKanpurPhy654/blob/main/Ising_model.ipynb

https://keras.io/api/datasets/

Keras

Search Keras documentation...

- About Keras
- Getting started
- Developer guides
- Keras 3 API documentation
  - Models API
  - Layers API
  - Callbacks API
  - Ops API
  - Optimizers
  - Metrics
  - Losses
  - Data loading
  - Built-in small datasets
    - MNIST digits classification dataset
    - CIFAR10 small images classification dataset
    - CIFAR100 small images classification dataset
    - IMDB movie review sentiment classification dataset
    - Reuters newswire classification dataset
    - Fashion MNIST dataset, an alternative to MNIST
    - California Housing price regression dataset

► Keras 3 API documentation / Datasets

Datasets

The `keras.datasets` module provide a few toy datasets (already-vectorized, in Numpy format) that can be used for debugging a model or creating simple code examples.

If you are looking for larger & more useful ready-to-use datasets, take a look at TensorFlow Datasets.

Available datasets

MNIST digits classification dataset

- load_data function

CIFAR10 small images classification dataset

- load_data function

CIFAR100 small images classification dataset

- load_data function

IMDB movie review sentiment classification dataset

- load_data function
- get_word_index function

Reuters newswire classification dataset

- load_data function
- get_word_index function

Fashion MNIST dataset, an alternative to MNIST

- load_data function

California Housing price regression dataset

- load_data function

It is possible to use these dataset to practice your ML skills

Multi-layer perceptron (MLP)

- Another name of Neural Net.
- Fully connected neurons with a nonlinear activation function.

CNN

Convolutional Neural Network (ConvNet)

CNN takes (mainly) images as input

We can train a usual DNN on images as well.

But, number of features become huge. As a result, number of weights is also huge.

Difficult to train even for a small image size.

Need to exploit spatial proximity of features.

Convolution (example of 2D image, aka grayscale image)

nxn

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6x6 image

fxf

f is usually odd number

$$* \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} =$$

3×3

Filter or Kernel

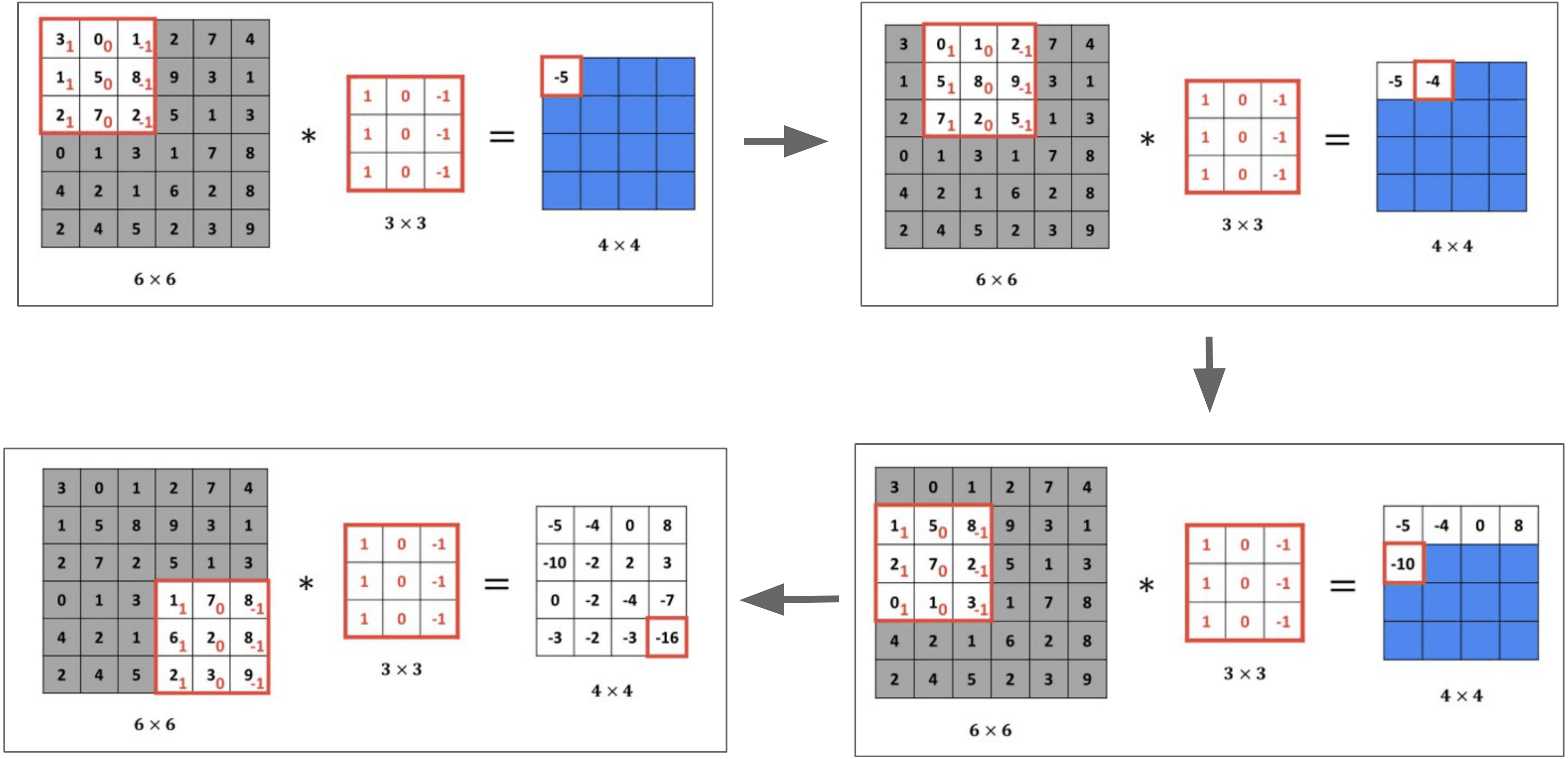
(n-f+1) x (n-f+1)

4x4 output

This filter can detect vertical edges in an image

Image shrinks

Convolution



A convolution operation converts all the pixels in its receptive field into a single value.

If you apply a convolution to an image, generally you will be decreasing the image size and bringing all the information in the receptive field together into a single pixel.

Vertical edge detector

2D image / grayscale