A workshop on: MATHEMATICAL ENGINEERING OF DEEP LEARNING - FOUNDATIONS

Convolutional Neural Networks
Presented by Benoit Liquet
University Of Pau, Pau, France

- Overview of Convolutional Neural Networks
- Filtering
- The Convolution Operation: VGG16
- Building a Convolutional Layer
  - Edge Detection
  - Can we use fully connected networks for image data?
  - Building a Convolutional Layer (Simplified)
  - Motivating a Convolutional Layer
  - Numbers in a Convolutional Layer
  - Parameters in a Convolutional Layer
  - A Convolutional Layer
  - Summary
- Building Convolutional Neural Networks
  - Pooling Layers
  - Fully-connected Layers
  - Activation Functions
  - Preprocessing
  - Weight Initialization
  - Regularization
  - Dropout
  - Batch Normalization
  - Understanding Kernel Size and Feature Maps
  - Hyperparameter Optimization
- Recent Developments
  - Learnable Activation Functions Constitute Neural Nodes
  - Fitted Convolutions
  - Convolutional Neural Networks Learn Translations
- Resource Recommendations

Based on Chapter 9 of the book Mathematical Engineering Deep Learning by Benoit Liquet, David Nott and Kim-Anh Le Cao

Slides are available at https://github.com/dnkim0/workshop-2

Overview of Convolutional Neural Networks

Convolutional neural networks are a family of neural network that specialize in processing grid-structured data, such as images (also, there is a strong local dependency in time series data).

Examples are images, text, and audio.

- Convolution is a specialized kind of linear operation.
- We will discuss later why specialized neural networks are computationally more efficient than fully connected neural networks. This is primarily because convolutional neural networks require fewer parameters than fully connected networks.
- Image data exhibits four key properties, namely, translation invariance and scarcity.

- **Translation invariance:** The classification decision for each image is **independent** of the position of the animal on the image, or out of an interpretation of where it appears on the image or the distribution of the image.

- **Locality:** The classification decision does not vary based on a given line in the image from the animal on the image. A cat is still a cat irrespective of whether the cats ears correspond to a building or a tree.

- Neural net when using fully fully connected neural networks for images, the first step is to **convert each image to a vector** before the **first layer**. By doing this we may lose both the above mentioned two structural properties.

Filtering

- Convolutional convolutional neural networks for helpful to have a basic understanding of filtering, a well understood tool for **signal processing**.
- Filtering is a method to process the **signals** (which, unlike the mathematical notion of a signal, is a function of time). Convolution is the **workhorse** behind filtering.
- Convolution is also a useful operation on input images, with the most common operation being the convolution. A convolution can be viewed as an operation between two functions, or two instances of functions, such as the **input** and the **kernel** or the **image** and the kernel or a **filter**.
- Each filter is a feature detector depending on the specific kernel at hand. For example, a popular filter, called the Sobel filter, is either a vertical edge detector or a horizontal edge detector, depending on the specific kernel at hand. Applying the Sobel filter to an image, followed by thresholding the outputs and then thresholding the resulting output, will yield the image shown on the upper right hand corner of the vertical edge detector.

Figure 9.1: Edge detection using the Sobel filter.

Note: Convolutional neural networks both open the classic idea of filtering using convolutional kernels. Each convolutional layer is made up of many **filters**, and each filter is a feature detector. Each filter produces a feature map for the input.

Example CNN Network: VGG16

- A good introduction to convolutional neural networks is to consider the **winning entry** from the ImageNet competition, using the VGG16 model, developed by Royal Societies Group, the group at Oxford University that created the network.
- The ImageNet competition is a competition for image classification, object localization, and object detection in images in 2014. It has 1.5 the number of images and classes, and over 2TB of data. It is a challenging task, and it is a competition.
- Top-5 error rate is 7.3% (relative to the megaflops, the output of 30 billion floating operations per second, or teraflop).
- The **VGG16** Network has over 138 million parameters.

Figure 9.2: The VGG16 Network Architecture.

Practice Q:
- Download 1 of CIFAR available on the website Github page. Alternatively, click here.
- Implement the VGG16 using Keras/PyTorch/TensorFlow.
- In this exercise, you can forget image augmentation, which are part of the ImageNet dataset, to see what is the output of a pretrained VGG16 model on the CIFAR-10 dataset. Pick 10 random images, resize them to a regular rectangular, and display your pretrained network.

The Convolution Operation

- As mentioned earlier, the convolution operation is a key component of convolutional neural networks.
- Convolution is an operation between two functions, or two instances of functions, such as the **input**, or **image**, and the **kernel**, or **filter**.
- The convolution operation between two matrices $\mathbf{X}$ and $\mathbf{K}$ is denoted by $\mathbf{X} * K$. Assume K_1, K_2 and M_1, M_2 are the dimensions of $\mathbf{K}$ and $\mathbf{X}$.
- In the context of CNN, $\mathbf{X}$ represents a tensor or input and $\mathbf{K}$ represents an input to the kernel. Further, it is given that $K_1 \leq M_1$ and $K_2 \leq M_2$.
- The convolution $\mathbf{S} = \mathbf{X} * \mathbf{K}$ is a tensor of dimension $(M_1 - K_1 + 1) \times (M_2 - K_2 + 1)$, where its value at i, j is:

$$S_{i,j} = (\mathbf{X} * \mathbf{K})_{i,j} = \sum_{m=1}^{K_1} \sum_{n=1}^{K_2} X_{i+m-1,j+n-1} K_{m,n}$$

- The convolution operation is illustrated below where $\mathbf{X}$ dimension 3×4 and a dimension 2×3. For $\mathbf{S}$ at $i = 0$ and $j = 0$:

$$S_{0,0} = (\mathbf{X} * \mathbf{K})_{0,0} = \sum_{m=1}^{2} \sum_{n=1}^{3} X_{m-1,n-1} K_{m,n}$$

Convolution operation

- In particular, the first element of the output matrix is computed as shown in the following figure.

Computing $S_{0,0}$. Note, it denotes element-wise product between matrices, not the matrix product, and $\sum$ denotes summation of all the elements of a matrix.

Edge Detection

- Let's consider the two dimensional presented earlier. We have seen that both horizontal and vertical edges are detected via convolution operation in which the filter/kernel.
- The filter $\mathbf{K}$ for detecting the horizontal edges is given by:

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

- The filter $\mathbf{K}$ for detecting the vertical edges is given by:

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

Note:
- The following operation of dark tops to LEFN is similar to convolut filtering, such as edge detection above. **However**, while classical filtering has been hand-designed, the CNN will learn the **kernel**.
- Applying **Sobel** filter and **Scharr** filter for filtering the filters using the kernel for image processing. Here, **Sobel** and **Scharr** are the **specific assumptions** made about the structure of the kernel. The problem is based only on computer vision detection.
- However, fully connected networks are inadequate when dealing with this task that high dim structured properties such as images.
- Convolutional neural networks are a family of neural network that specialize in processing grid-structured data, such as images (also, there is a strong local **translation invariance** in time series data).
- As a result, the number of parameters to learn in convolutional neural networks is significantly smaller than in corresponding fully connected neural networks.

Can we use fully connected networks for image data?

Practice Q:
- Use the fully connected neural network to image in a dimension $\mathbf{X}_{i,j}^{(0)}, \mathbf{X}_{i,j}^{(1)}$ and map that connected to a vector of length $\mathbf{X}_{i,j}^{(2)}$. Use 100 in a hidden layer.
- Minimize loss and generate we can continue to train the elements of the vector $\mathbf{X}$ in the network $\{\beta, \beta^{(1)}, \ldots, \beta^{(L)}, \ldots, \mathbf{X}_{i,j}^{(L)}\}$.
- The following figure shows needs to vectors connected before inputting it to a single layer.

A convolution needs to vector connected before inputting it to a single layer.

- The following figure shows transformation of the needs from the corresponding vector. This operation can be seen as a kind of a fully connected layer. The length of the output is a product of the integers.

Reshaping the tensor into the vector.

Issues:
1. The input both the structural properties: translation invariance and locality.
2. The many parameters to learn.

Motivating a Convolutional Layer (Simplified)

- Let's consider a simplified version of a convolutional layer.
- Assume a single input channel, and a single output channel, and no **spatial structure**. In convolutional layers, we build a feature that are looks at local regions of the image, **processing small area at a time**. The kernel filters take an operation **neighbor** of pixels, and **not** the entire image, and **not** the entire image at a time.

Reducing Parameters with Spatial Regularity
- Suppose we have a 1000 x 1000 image, and we are using the **simplified** version of the layer. The filter is a small window that **slides over the image**, examining one small region at a time, looking at **local features** needed to do **edge** detection, **blob** and **corners**, **lines**, **T**, **L**, **X**, **Y**, **Y**, **Y

Solution Practice Exercise 4

- Solution Practice 2
- Solution Practice 3

Practice Exercise

Practice 5

- Open Tutorial 3 of CNN available on the workshop GitHub page. Alternatively, click here.
- Save a copy of this in your Google Drive.
- In this exercise, we build a simple CNN for classification using MNIST dataset.

Solution Practice Exercise

Practice 5

- Open Tutorial 3 of CNN available on the workshop GitHub page. Alternatively, click here.
- Save a copy of this in your Google Drive.
- In this exercise, we build a simple CNN for classification using MNIST dataset.

Other Landmark Architectures

In addition to VGG19 architectures that achieved high accuracy in the **ImageNet Large Scale Visual Recognition Challenge (ILSRVC)** 2014, there are few other landmark architectures. The following list is a small summary of these models:

- **ResNet** (Residual Network): ResNet is a CNN architecture that **introduced residual blocks**, which allow for the reuse of feature maps from earlier layers by adding skip connections. This enables very deep networks (e.g. 152 layers) to be trained and achieve state-of-the-art performance on image classification tasks.

A shortcut connection (residual connection) as part of a residual network.

- **Inception** (GoogLeNet): Inception is a CNN architecture that introduced the Inception module, which uses multiple filter sizes in parallel to extract features at different scales. This allows for efficient use of computational resources and achieves high accuracy on image classification tasks.

One form of an inception module, playing part in an inception network. The key idea is parallel computation of various parts followed by a concatenation of the outputs for the next layer.

- The inception network is a type of convolutional neural network architecture that was introduced in **2014 by Google researchers**. The key idea of the inception network is to use multiple filter sizes in a single layer of the network, in order to capture features at different scales and resolutions.
- This approach is different from other popular CNN architectures, such as VGG, which typically use small 3x3 filters in all of their convolutional layers.
- The inception network also includes the use of so-called bottleneck layers, which use 1x1 convolutions to reduce the dimensionality of the feature maps before applying larger filters. This helps to reduce the number of parameters in the network, while still allowing it to capture complex features.
- One of the key innovations of the inception network is its use of an inception module, which is a small network that combines the various filter sizes and bottleneck layers. The inception module allows the network to capture complex and multi-scale features at each layer, while also reducing the number of parameters in the network.
- **MobileNet**: MobileNet is a CNN architecture designed for mobile and embedded devices that prioritizes lightweight computation to reduce the number of parameters and computations required while maintaining high accuracy. It achieves **state-of-the-art performance on mobile platforms and real-time video analysis**.
- **DenseNet**: DenseNet is a CNN architecture that introduces dense connectivity between layers, where each layer receives feature maps from all previous layers. This promotes feature reuse and reduces the number of parameters required, leading to high accuracy and efficient training.
- **AlexNet**: AlexNet is a CNN architecture that was the winner of the **ILSRVC 2012**. At the time, it was a breakthrough in image classification, demonstrating state-of-the-art performance on the ImageNet dataset. It introduced the concept of using **ReLU** activation functions, **dropout regularization**, and **data augmentation** for training deep neural networks. It also used GPU acceleration to speed up training and achieved a significant reduction in error rate on image classification tasks.
- **EfficientNet**: EfficientNet is a family of convolutional neural network architectures that were developed with the aim of providing high accuracy while being computationally efficient and scalable. The key idea of EfficientNet is to use a combination of dimensions of the network parameters such as depth, width, and resolution in a balanced way, while also introducing a new compound scaling method that optimizes these dimensions based on a set of pre-defined constraints.

Accuracy-efficiency trade-off of various convolutional models as well as efficient net

Beyond Clasation

- So far we have focused on the interests of convolutional neural networks with a particular emphasis on the task of **image classification**, i.e. determining if an image is that of a cat or a dog.
- However, there are several other important image analysis tasks that are also heavily addressed with convolutional neural networks. These tasks deal with analysis and understanding of an image including the location of objects, the count of objects, separating between different semantic features of the image, and more.

Note: In some of the slide ahead, it is important to keep in mind that not all data is made of monochrome or color images. Within computer vision, one other class with **image sequences** (short movies), or images that have **more than 3 channels**.

- For example, some imaging modalities also have a **distance channel** capturing the distance from the camera per pixel. Further, non-image data can also be handled via convolutional networks.
- One such example is **MRI (functional magnetic resonance imaging)** data which is 4-dimensional in nature as it records the state of physical locations (e.g. blood oxygen level-dependent (BOLD) signal) in three dimensions over time.

Convolutional Networks and Key Computer Vision Tasks

- As mentioned above, classification serves as a simple and useful example. For a given input image, a convolutional neural network (CNN) can be used to predict $p(y \mid x)$ which is a **vector of probabilities** where the highest probability typically determines the appropriate label for the image.
- An area evident from our detailed study of the VGG19 model and other architectures, **initial layers** of the model, $f_\phi(\cdot)$ are typically **convolutional**, and the final layers are typically **fully connected** layers.
- These final layers **help** transition the internal dense feature maps in the network into the output vector of probabilities $\hat{y}$.
- Let us denote the output of the convolutional part as h (a vector) and denote the fully connected layers as $g_\theta(\cdot)$. Then, we can express the entire network as $g_\theta(f_\phi(x))$.
- The following figure illustrates key computer vision tasks for images.

Figure 8.15: Illustration of some common computer vision tasks involving images. (a) Object classification. (b) Object detection. (c) Object detection. (d) Semantic segmentation. (e) Instance segmentation. (f) Monocular depth estimation. (Data: Kanopy).

- **Object localization** which is the task of identifying the location of an object in an image, as well as possibly the type of object in which case the task is called **localization and classification**.
- **Object detection** which is the task of detecting multiple instances of an object in an image, also separating between the objects and classifying their type.
- **Landmark detection**, which is the task of identifying the specific local locations of landmarks in an image.
- **Semantic segmentation**, which is the task of classifying each individual pixel in a given image into a fixed set of classes (pixel level classification).
- **Instance segmentation**, which finds different instances of objects in the image and separates pixels to be of different instances.
- **Identification** (face recognition), which determines if an image is that of a specific instance (or person).

Let us now consider possible formulations of these tasks:

- For object localization, f needs to contain information about a **bounding box** which locates the object. This can be in the form of $(\hat{l}_x, \hat{l}_y, \hat{l}_w, \hat{l}_h)$ where $\hat{l}_x$ and $\hat{l}_y$ are the coordinates of top left corner of the bounding box and $\hat{l}_w$, $\hat{l}_h$ are the height and width of the bounding box, respectively.
- For object detection, a collection of multiple bounding boxes needs to be supplied.
- For landmark detection, a list of coordinates of the locations of landmarks comprises the output.
- For semantic segmentation, each pixel location in the input image, i, has an associated probability vector of classes in the output $\hat{y}$. Hence in this case, $\hat{y}$ can be represented as a tensor with width and height dimensions the same as the input image, and a depth dimension equal to the number of classes, C, in the classification problem.
- For instance segmentation, the output is similar to that of semantic segmentation, but instead of receiving probabilities of classes, the depth dimension of the output $\hat{y}$ is used for determining the specific instance of any given pixel.
- Finally, in the case of identification, or face recognition, the output $\hat{y}$ can be a probability or a binary classifier, since the task is to determine if a face image matches a given pre-stored template or not. The specialty in this case is that the **input** x is typically compared against a database of there one is the template of the person (e.g. a stored image in a security database), and the other image is the **current image taken**.

Object Localization

- To get a feel for object localization assume that we want to look a CNN that **answers on an input image** x and determines if the image contains a dog or a cat (a pure classification).
- In addition to classification, we also want to determine the **specific location** $(\hat{l}_x, \hat{l}_y, \hat{l}_w, \hat{l}_h)$ of that object (localization). Images with multiple items or objects are not considered. Images without a bird and without a plane are possible and in this case focus lies on the cat and dog pixels surrounding.

One way to encode the output is

$$\hat{y} = [\hat{p}_{cat}, \hat{p}_{dog}, \hat{p}_{bird}, \hat{p}_{plane}, \hat{l}_x, \hat{l}_y, \hat{l}_w, \hat{l}_h],$$

where an $\hat{}$ standard classification example ($\hat{p}_{cat}$, $\hat{p}_{dog}$, $\hat{p}_{bird}$, $\hat{p}_{plane}$) is probability vector, and the other coordinates define a bounding box.

- Here we output the $\hat{p}_{cat}$, $\hat{p}_{dog}$, greater than each of $\hat{p}_{bird}$ and $\hat{p}_{plane}$ implies a prediction of cat or dog.
- In terms of training data, for each input image we denote the output as y where images without a bird or a plane are labeled as, $y = [0, 0, 0, 0, \bullet, \bullet, \bullet, \bullet]$, where $\bullet$ are "don't care" values. Images with a cat or a dog are labeled as $y = [0, 1, 0, 0, l_x, l_y, l_w, l_h]$, where the bounding box (l_x, l_y, l_w, l_h) is labeled based on a manual determination by a human annotator. Similarly, images with a plane are labeled as $y = [0, 0, 1, 0, l_x, l_y, l_w, l_h]$.
- We now can define a loss function that optimizes between $\hat{y}$ and y. For this we first separate the classification and localization objectives in the loss function as $\mathcal{L}(\hat{y}, y) = \mathcal{L}_{class}(\hat{p}, p) + \mathcal{L}_{box}(\hat{l}, l)$. The former depends only on the probability components $\hat{p}$ and p, and the latter depends only on the bounding box components $\hat{l}$ and l. For the classification loss, we use categorical cross-entropy. For the localization loss, we use a mean squared error, applied to the four bounding box components.

This loss separate losses are then combined such that the loss for a specific observation is:

$$\mathcal{L}_{classification}(\hat{p}, p) + (I \cdot \alpha) \cdot \mathcal{L}_{box}(\hat{l}, l),$$

where I - if is a ground-truth parameter used to weigh the two losses and taken as $I = 1$ by default. Observe that α is a hyper-parameter to be tuned, and is interested 5 and thus for all labels in the training data without a bird or a plane only the classification objective is used.

- To perform object localization, say with a model like **VGG19**, the network can be modified by adding additional layers at the end of the network to output the bounding box coordinates. Typically, these layers are fully connected layers that take the output of the last convolutional layer of the network. The expansion head consists of fully connected layers that predict the coordinates of the bounding box. Such simple modifications of networks that were otherwise designed for classification are always possible.

Practice Exercise

Practice 4

- Open Tutorial 3 of CNN available on the workshop GitHub page. Alternatively, click here.
- Save a copy of this in your Google Drive.
- In this exercise, we build a simple CNN for classification using MNIST dataset.

Solution Practice Exercise

- Solution Practice 2
- Solution Practice 3