Deep Learning and Neural Networks

Demetrio Labate

March 18, 2024

Part 3
Beyond CNNs

3.1 Limitations of CNN models

Limitations of CNN models

There are notable drawbacks of CNN models that are very critical in some applications

1. classification of images with different viewpoints or illuminations;
2. classification of scrambled images;
3. coordinate frame;
4. interpretability, black box problem;
5. adversarial examples;
6. computational complexity and need for many training samples.

Limitations of CNN models - image viewpoint/illuminations

One major challenge in computer vision is to deal with the variance or real world data, specifically how to identify imaged objects

- Under different angles;
- Under different backgrounds;
- Under different lighting conditions.

Limitations of CNN models - image viewpoint/illuminations

CNNs have achieved great performance while classifying images which are very similar to the dataset on which they are trained.

However, if images contain some degree of tilt or rotation, or some change in illumination with respect to the training samples, then CNNs usually have difficulty in classifying the image.

Note that images in ImageNet and the other common datasets are collected with consistent illumination and orientation.

This challenge can be solved by adding different variations to the image during the training process otherwise known as **Data Augmentation**.

Data augmentation

This technique is often adopted to increase the amount of available data and **avoid the overfitting issue**. It is also useful to improve the classification performance of a classification model, with respect to variations in the data due to changes in viewpoint, background and illumination.

Data augmentation aims to improve the attributes and size of training datasets by incorporating a collection of methods such as:

1. **Geometric transformations**: randomly flip, crop, rotate, stretch, and zoom images.
2. **Color space transformations:** randomly change RGB color channels, contrast, and brightness.
3. **Kernel filters:** randomly change the sharpness or blurring of the image.
4. **Random erasing:** delete some part of the initial image.
5. **Mixing images:** blending and mixing multiple images.
6. **Noise injection:** adding random noise to the image.

Data augmentation

Geometric transformations include several possible operations. You need to be careful about applying multiple transformations on the same data, as this can reduce model performance.

- **Flipping** an image along the vertical or horizontal axis is highly simple to implement and has been verified as valuable on datasets like ImageNet and CIFAR-10.
- **Rotation.** Rotated images within 0 to 360 degrees around the center are easily generated. Not every rotation is suitable. For instance, in digit recognition tasks, small rotations (from 0 to 20 degrees) are helpful but larger rotations are not as data labels cannot be preserved for large rotation degrees.
- **Translation.** To avoid positional bias within the image data, a useful transformation is to shift the image up, down, left, or right. When translating the initial images in a particular direction, the residual space should be filled with random noise or a constant value so that the spatial dimensions of the image post-augmentation are preserved.

Data augmentation

image flip

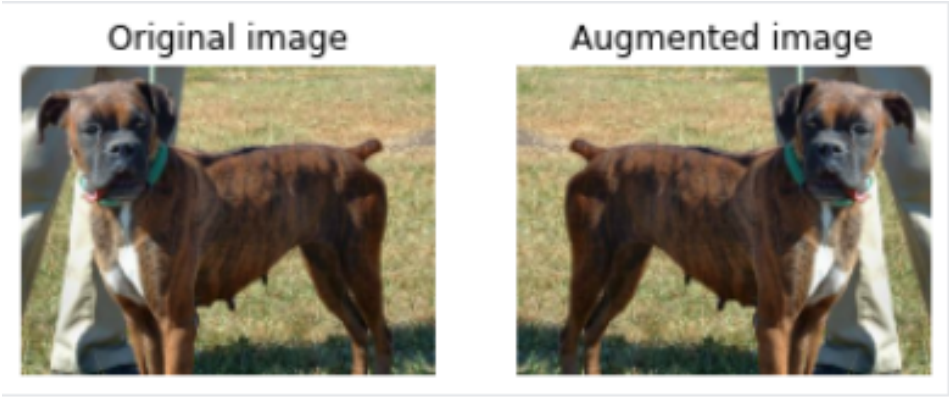
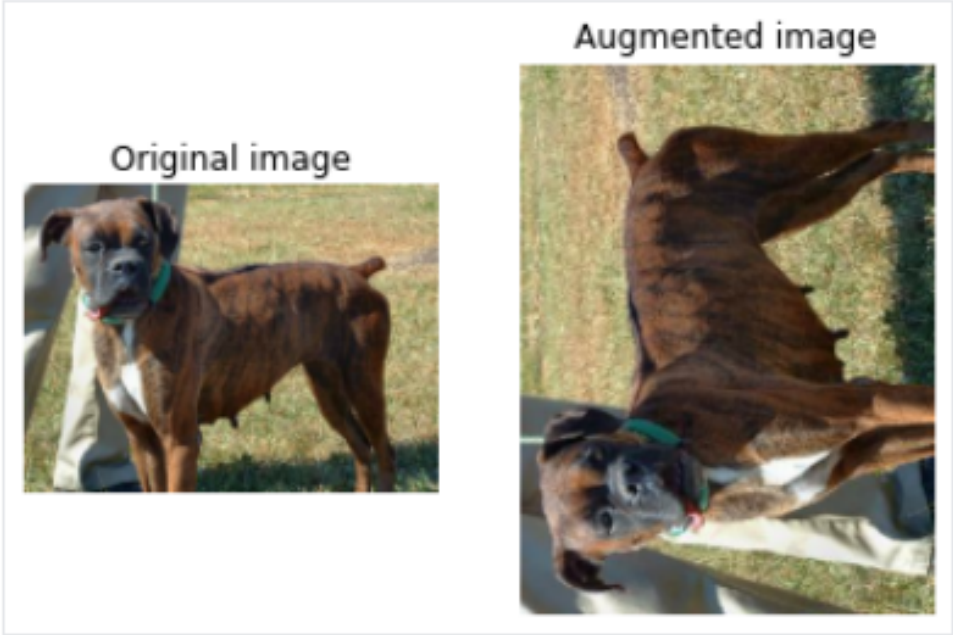


image rotation



Data augmentation

Color space transformations

- **Color channel transformations.** An easy color augmentation involves separating a channel of a particular color, such as Red, Green, or Blue. To convert an image using a single-color channel it is sufficient to insert additional double zeros from the remaining two color channels.
- **Contrast, brightness, saturation** can be Increased or decreased in an image using well established matrix operations to manipulate the RGB values, similar to common photo-editing applications.

Cropping. Data can be augmented by cropping a dominant patch of an image. Random cropping may be employed to produce an impact similar to translations. The difference between translations and random cropping is that translations conserve the spatial dimensions of this image, while random cropping reduces the input size.

Data augmentation

image brightness

Data augmentation

image saturation

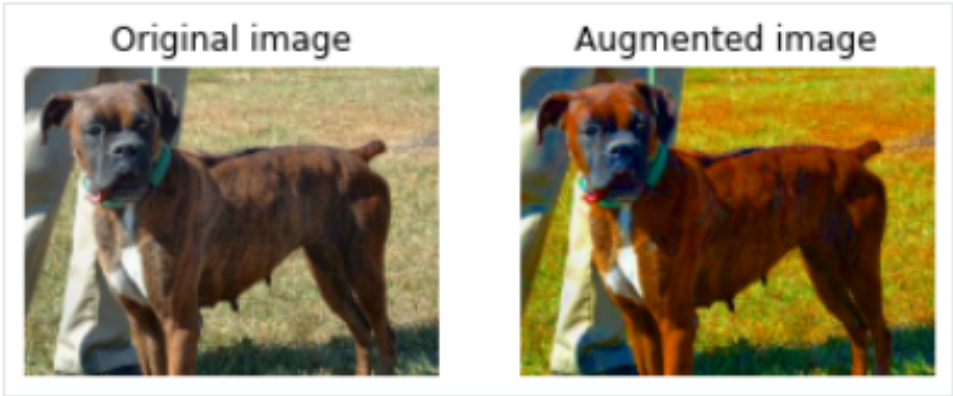


image crop

Data augmentation in PyTorch

Pytorch has a module available called

`torchvision.transforms`

that lets us augment images in different ways.

You can add transform layers within `torch.nn.Sequential` or apply an augmentation function separately on the dataset.

Here is a tutorial to data augmentation in PyTorch.

Augmentor is also a Python package for image augmentation and artificial image generation. You can perform Perspective Skewing, Elastic Distortions, Rotating, Shearing, Cropping, and Mirroring. Augmentor also comes with basic image pre-processing functionality.

Data augmentation

Remarks on data augmentation.

- Data augmentation techniques are not limited to images. You can augment audio, video, text, and other types of data.
- One need to be careful in choosing the appropriate data transformations for data augmentation to ensure that they do not change the image label. For instance, rotations of the digit 9 and 6 may change their label. Similarly, arbitrary cropping and flipping may result in chaging the object label.
- The main disadvantage of data augmentation arises from data bias. If the original data have biases, the augmented data will also have biases which will lead to suboptimal results.

Limitations of CNN models - Scrambled Images

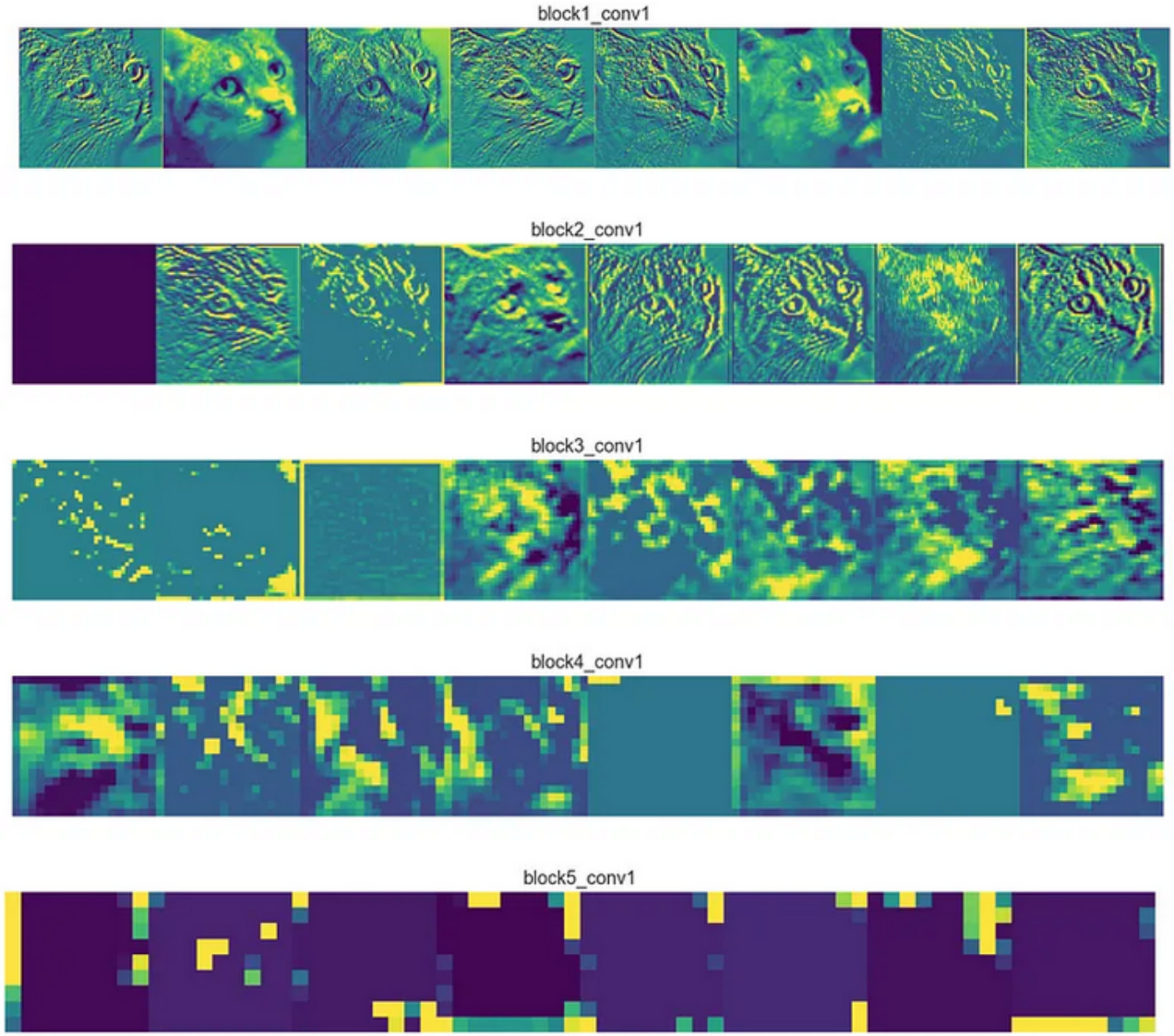
While CNN are very efficient to extract image features, they are not meant to preserve hierarchical structures and spatial relationships of the features in the data.

In a CNN, early layers extract simple and interpretable features such as edges,corners and endpoints.

Higher level layers in the CNN combine extracted features regardless of order.

As network gets deeper, **features become more abstract** and shrink in size **due to repeated pooling and filtering**.

Limitations of CNN models - Scrambled Images



As you progress through the layers, feature maps become increasingly complex and abstract.

Limitations of CNN models - Scrambled Images

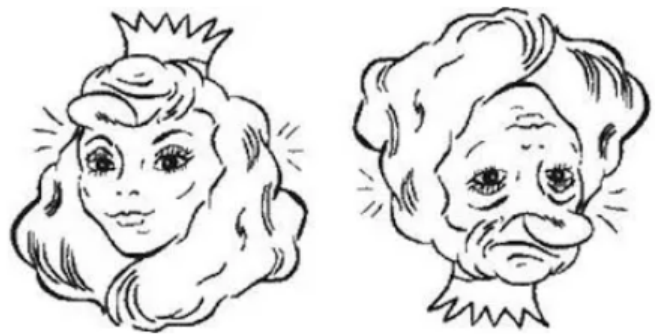
A CNN does not keep track of spatial relationships.

To a CNN the above pictures appear very similar as both contain the same features.
This is also called the *Picasso problem* in image recognition.

Limitations of CNN models - Coordinate frames

CNNs do not have **coordinate frames**, that is, they lack an internal representations of components and their part-whole relationships.

A coordinate frame is a mental model which keeps track of the orientation and different features of an object.



If we look at the figure we can identify that the image on the right, if turned upside-down will give us the image on the left. Just by mentally adjusting our coordinate frame in the brain we are able to see both faces, irrespective of the picture's orientation.
The human Coordinate frame enables us to see both the faces.

Limitations of CNN models - Black box problem

In a seminal paper titled "*Visualizing and Understanding Convolutional Neural Networks*" (2013), Zeiler and Fergus begin by observing that the success of CNNs is due in large part to the accessibility of large training sets and increased computational power with the usage of GPUs.

They point out to the limited knowledge that researchers have on inner mechanisms of these models.

in other words, a CNN is a **black box** for which "development of better models is reduced to trial and error".
While we do currently have a significantly better understanding, this still remains an issue.

One main contributions of their paper was a way of **visualizing feature maps**.

Limitations of CNN models - Black box problem

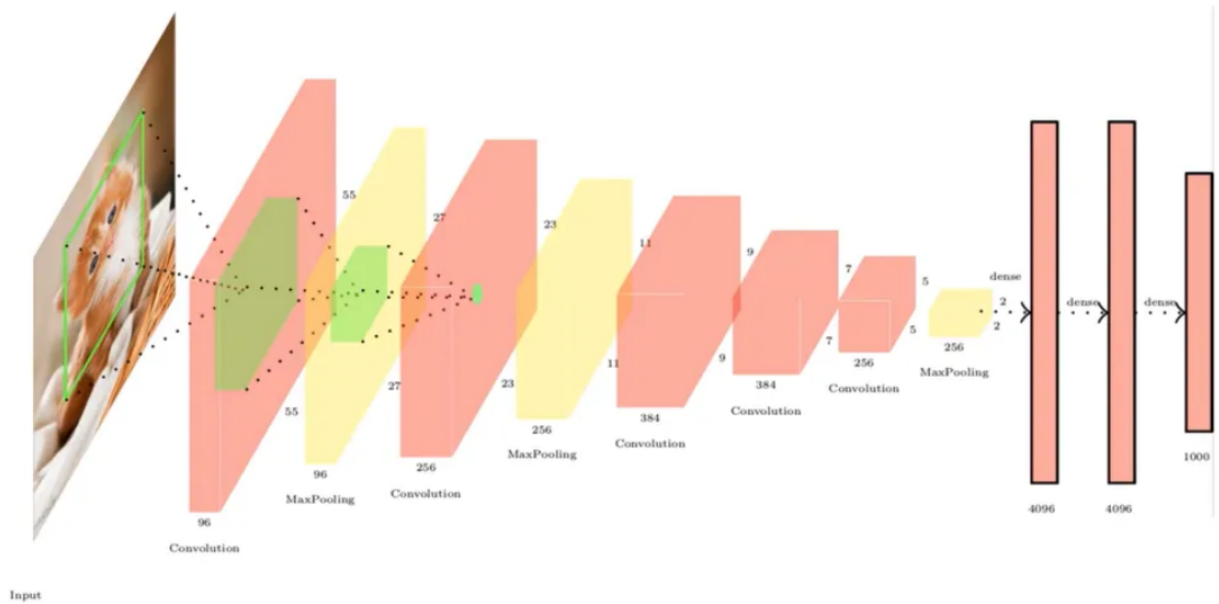
Questions:

- How is CNN model is learning the complex dependencies present in the input?
- What kind of data/images cause certain neurons to fire?
- How good are the hidden representations of the input image?

To answer these questions, we can **visualize the learned filter weights**

Limitations of CNN models - Black box problem

The **receptive field** of a neuron is the region in the input image that can influence the neuron in a convolution layer, that is, how many pixels in the original image are influencing the neuron present in a convolution layer.



As we go deeper in the CNN, the pixels at the deeper layers will have a high receptive field, that is, the region of interest with respect to the original image would be larger.

Limitations of CNN models - Black box problem

Understanding the receptive field is important as it gives us an idea of where we are getting our results from as data flows through the layers of the network.

We are interested in the size of the receptive field in our initial input to understand how much area the CNN covers from it.

This is essential in many computer vision tasks.

Consider, for example, the task of object detection. The network takes an input image and predicts the class label of every pixel building a label map in the process. If the network does not have the capacity to take into account enough surrounding pixels when doing its predictions some larger objects might be not fully contained in the receptive field.

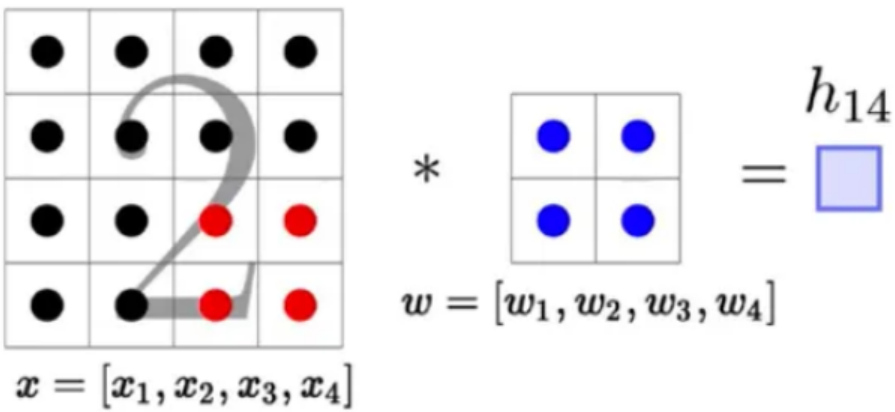
Computing Receptive Fields

Computing Receptive Fields - short version

Limitations of CNN models - Black box problem

Filter Visualization. By visualizing the filters of the trained model, we can understand how CNN learns the spatial pixel dependencies present in the image.

Consider that we have a 2D input of size 4x4 and we are applying a filter of 2x2 (in red) on the image starting from the top left corner of the image. As we slide the kernel over the image from left to right and top to bottom to perform a convolution operation we would get an output that is smaller than the size of the input.



Limitations of CNN models - Black box problem

The output at a convolution operation is equal to the dot product of the input vector and a weight vector. We know that the dot product between the two vectors is proportional to the cosine of the angle between vectors.

During convolution operation, certain parts of the input image might give high value when we apply a filter.

The output would be high if the cosine value between the vectors is close to 1.
That means **the neuron is going to fire maximally when both input vector (portion of the image) and the weight vector are in the same direction**.

Limitations of CNN models - Black box problem

Conclusion

- We can think of a convolutional kernel or filter as an image.
- As we slide the filter over the input from left to right and top to bottom whenever the filter coincides with a similar portion of the input, the neuron will fire.
- For all other parts of the input image that does not align with the filter, the output will be low.
- The kernel or weight matrix is called a filter because it filters out portions of the input image that do not align with the filter.

Limitations of CNN models - Black box problem

We will visualize the trained filters of AlexNet.

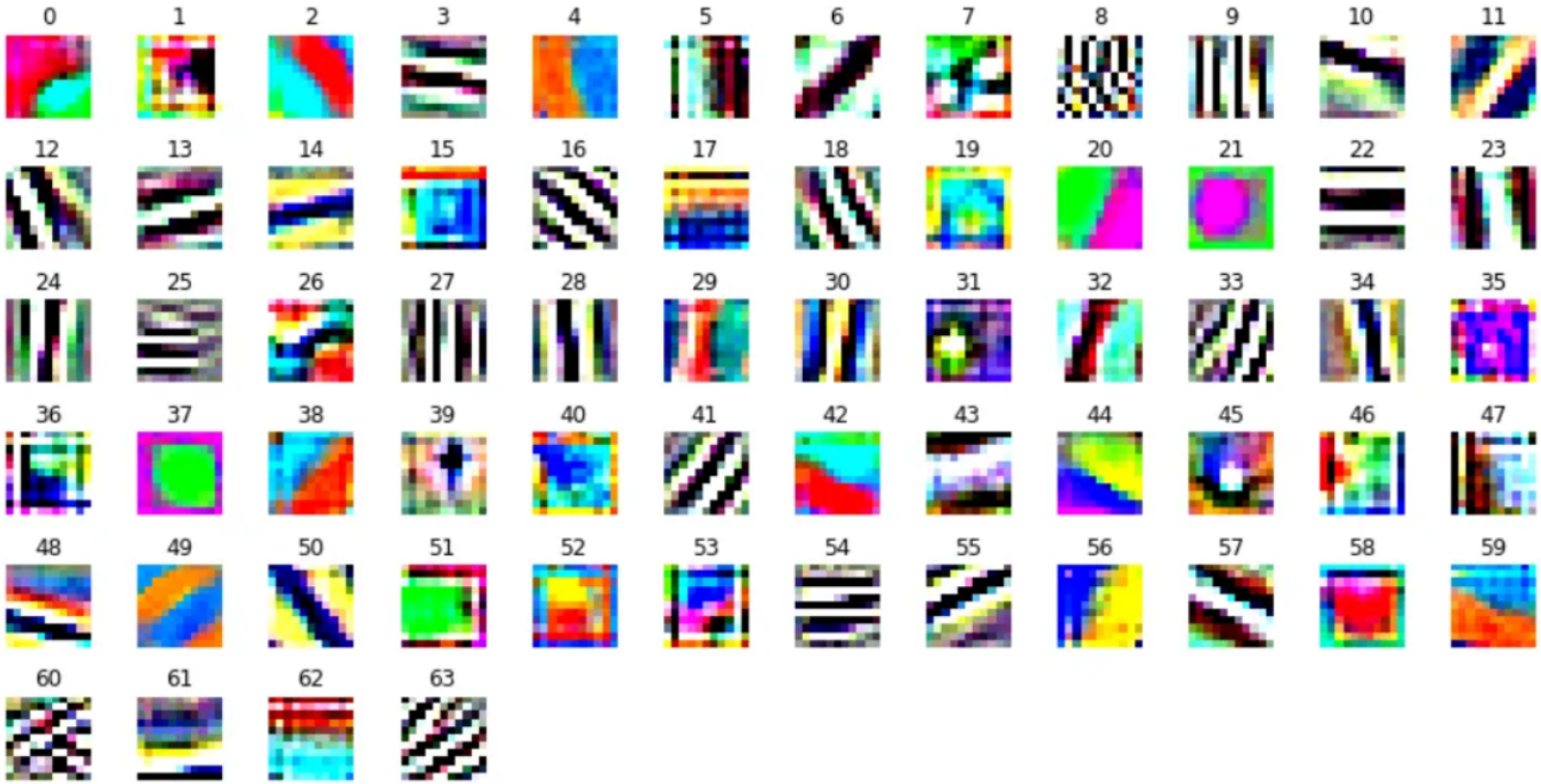
Recall that this network contains 5 convolutional layers and 3 fully connected layers. ReLU is applied after every convolution operation. Remember that in convolution operation for 3D (RGB) images, there is no movement of kernel along with the depth since both kernel and image are of the same depth.

We will visualize these filters (kernel) in two ways

1. By combing three channels as an RGB image.
2. By visualizing each channel in a filter independently using a heatmap.

Limitations of CNN models - Black box problem

We have 64 filters of depth 3 (RGB) in the first convolutional layer. We combine each filter RGB channels into one RGB image of size 11x11x3.



Limitations of CNN models - Black box problem

Interpretation

From the images, we see that the kernels seem to learn blurry **edges, contours, boundaries**.

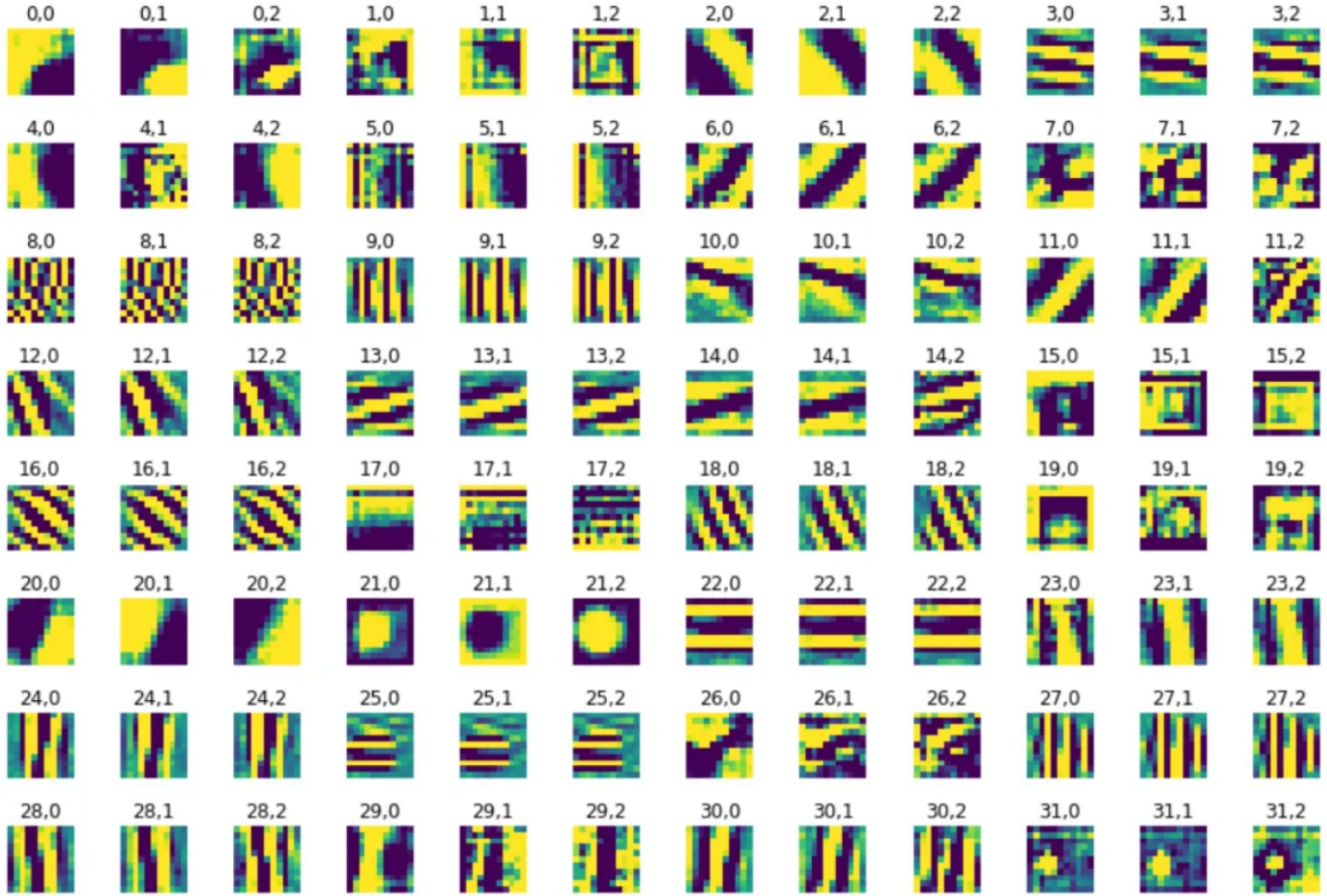
For example, figures 4, 20, 42, 59 in the above image indicate that the filters are trying to learn a edges in different orientations.

Figures 31, 37 indicate that the filters have learned some contours.

Figures 3, 8, 9, 12, 16 and more show periodic patterns of different frequency and orientation.

Limitations of CNN models - Black box problem

Alternatively, we can display 3 separate images for each channel since the depth of the filter is 3. In total, we now have 64*3 images as the output for visualization.



Limitations of CNN models - Black box problem

By looking closely at the filter visualizations above, it is clear that the patterns found in some of the channels from the same filter are different. That means not all channels present in a filter are trying to learn the same information from the input image.

Deeper layers:
As we go deeper and deeper into the network, the number of filters used for convolution increases.

The second convolution layer of Alexnet has 192 filters, so we would get 192*64 = 12,288 individual filter channel plots.

It is not practical to visualize all these filter channels individually either as a single image or each channel separately because of the large number of such filters. A way to plot these filters is to concatenate all the images into a single heatmap using greyscale.

As we go deeper into the network it becomes harder to interpret the filters.

Limitations of CNN models - Black box problem

Visualizing Convolution Neural Networks [blog]

"Visualizing and Understanding Convolutional Networks" by Zeiler and Fergus, 2013 [paper link]

Limitations of CNN models

Geoffrey Hinton attributed most CNN drawbacks primarily to the use of **pooling**, which is employed to down-sample spatial information.

Although effective in expanding the network's field of vision and retrieving high-level characteristics, **the pooling operation discards important information about hierarchical structures and geographical relationships in the image**.

Hinton: *"The pooling operation used in CNNs is a big mistake and the fact that it works so well is a disaster"*

Limitations of CNN models

Drawbacks of pooling:

- It violates biological shape perception in that it has no intrinsic coordinate frame.
- It provides (local) shift-invariance (positional information is discarded) **instead of shift-equivariance** (shifted input is mapped to shifted output).
- It ignores the linear variations that underlies many variations among images;
- It routes statically instead of communicating a potential "find" to the feature that can appreciate it;
- It damages nearby feature detectors, by deleting the information they rely upon.

These drawbacks were taken into consideration and led Hinton to the idea of **Capsule Neural Networks** or **CapsNets** [Sabour, Frosst, Hinton, 2017].

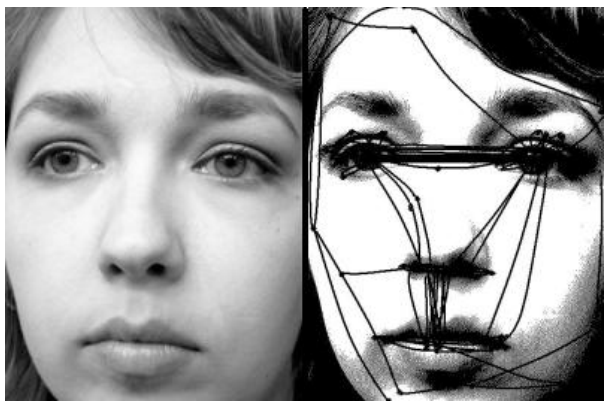
CapsNets

CapsNets were introduce to overcome the limitations of CNNs related to learning hierarchical structures and spatial relationships.

CapsNets are designed to emulate **hierarchical relationships**, drawing inspiration from biological neural systems.

Visual fixation and saccades. [Video Lecture on Eye Movements]

- Human vision uses **saccades**, that is quick, simultaneous movement of both eyes between two or more phases of **fixation** to scan a scene.

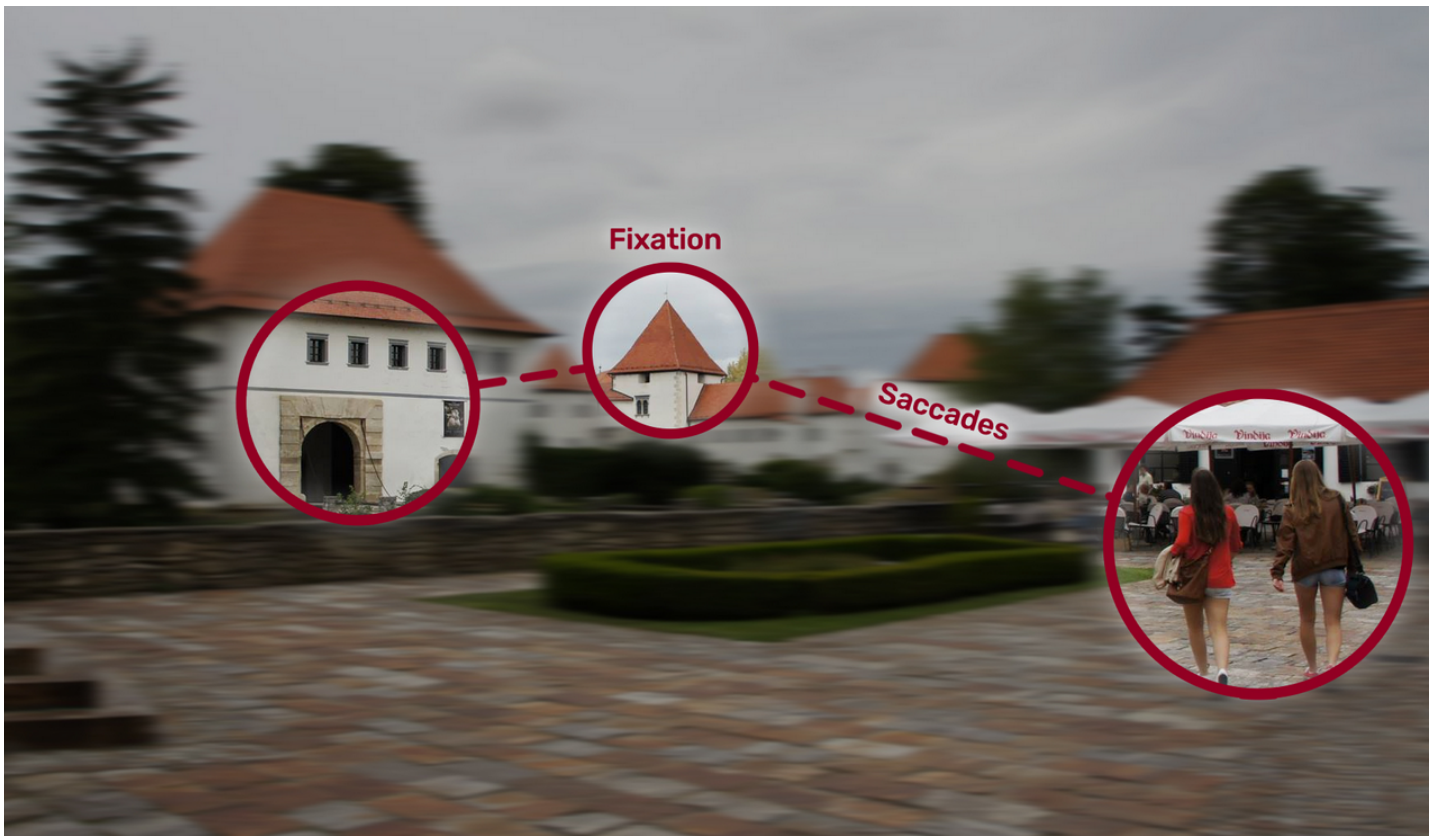


Trace of saccades of the human eye on a face while scanning

CapsNets

Using fixation and saccades, human vision:

- ignores irrelevant details by a careful sequence of fixations;
- only a tiny fraction of the optic array is processed at the highest resolution;
- this multilayer visual approach creates a **parse tree** on each fixation, capturing the hierarchical organization of the scene.



CapsNets

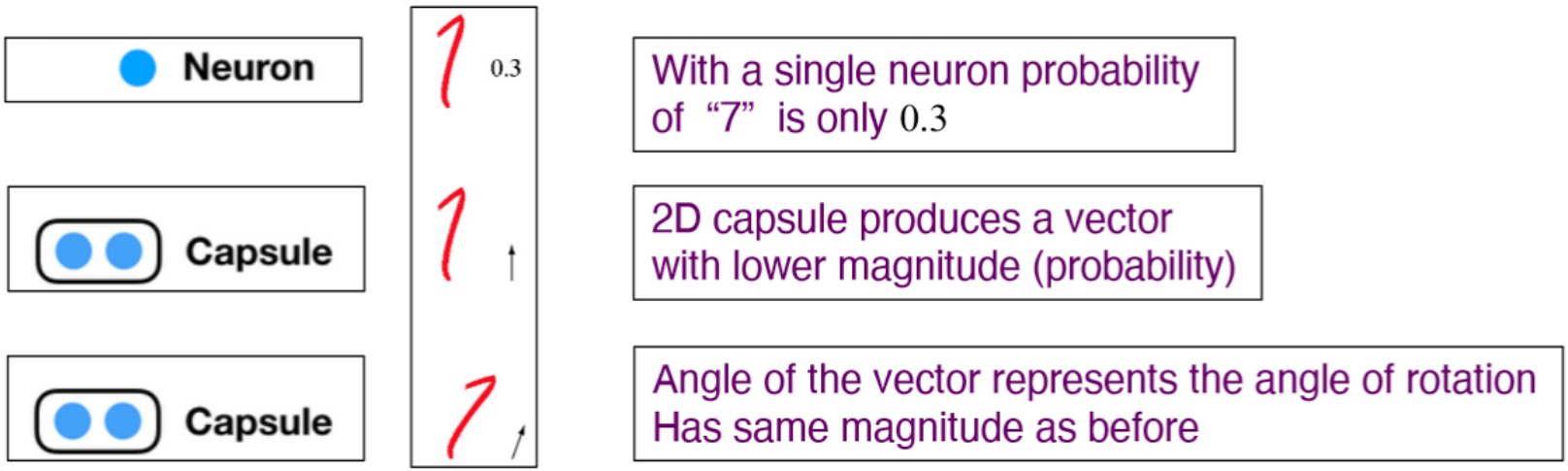
The fundamental building block of a CapsNet is called a **capsule.**

- A capsule is a group of neurons whose outputs represent different properties of the same entity.

- Capsules are designed to handle hierarchical structures and pose variations by encapsulating both the activation information and spatial relationships.

CapsNets

- Each capsule outputs a set of pose parameters, including orientation and position, in addition to an activation that represents a particular entity or portion of an object.



- Unlike neurons that work with scalars, capsules process inputs by encapsulating the result in informative **vectors** through an affine transformation.

CapsNets

While neurons in classical neural networks are associated with pointwise activation, sum computation and weighted connections, capsules go through more steps:

1. Input vectors multiply with spatial-relationship-encoded weight matrices;
2. further multiplication with weights;
3. weighted sum of input vectors;
4. activation function application for vector output.

CasNets iteratively refine the coupling coefficients between capsules based on the agreement of their pose parameters, capsules enabling **dynamic routing**.

- **1. Input vectors multiply with spatial-relationship-encoded weight matrices.**
  Input vectors (either the initial input or information from a preceding layer) are transformed via multiplication with weight matrices.
  These weight matrices encode spatial relationships within the data. For instance, if one object is positioned symmetrically around another and shares similar proportions, the resulting product of the input vector and weight matrix encapsulates a high-level feature indicative of this spatial arrangement.
  This process allows the neural network to capture meaningful relationships and features as it progresses through its layers.
  Here, input vector are multiplied by the weight matrix.

CapsNets

- **2. Further multiplication with weights.**
  A weighted adjustment in a capsule network is applied to the outputs obtained from the previous step. In contrast to conventional neural networks which update weights via error-based backpropagation, Capsule Networks use **dynamic routing to change weights**.
  CapsNets create robust associations between nearby high-level and low-level capsules by dynamically adjusting weights.
  The computation entails calculating the separation between dense clusters that represent low-level capsule predictions and the outputs of the affine transformation. When low-level capsule predictions are similar, these clusters form and are positioned closely. The distance metric determines the weight assignment so that the high-level capsule closest to the current prediction cluster is given a higher weight and the other capsules have lower weights according to their distances.

- **3. 3. Weighted sum of input vectors**
  Calculate the input vectors' weighted sum.
- **4. Activation function application for vector output.**
  Capsule activation functions guarantee dynamic representations of vector outputs. The **squashing function**, which normalizes the vector length while maintaining its orientation, is given by

$$v_j = \frac{||s_j||^2}{1 + ||s_j||^2} \frac{s_j}{||s_j||}$$

  Where, s_j is the sum of the input vectors and v_j is the output obtained after applying non-linearity function.
  By compressing the vector s_j to a magnitude between 0 and 1, enables robust hierarchical representations by capturing intricate interactions among features. Capsules can transmit subtle information that is essential for complex pattern identification because of the normalizing property of the squashing function.

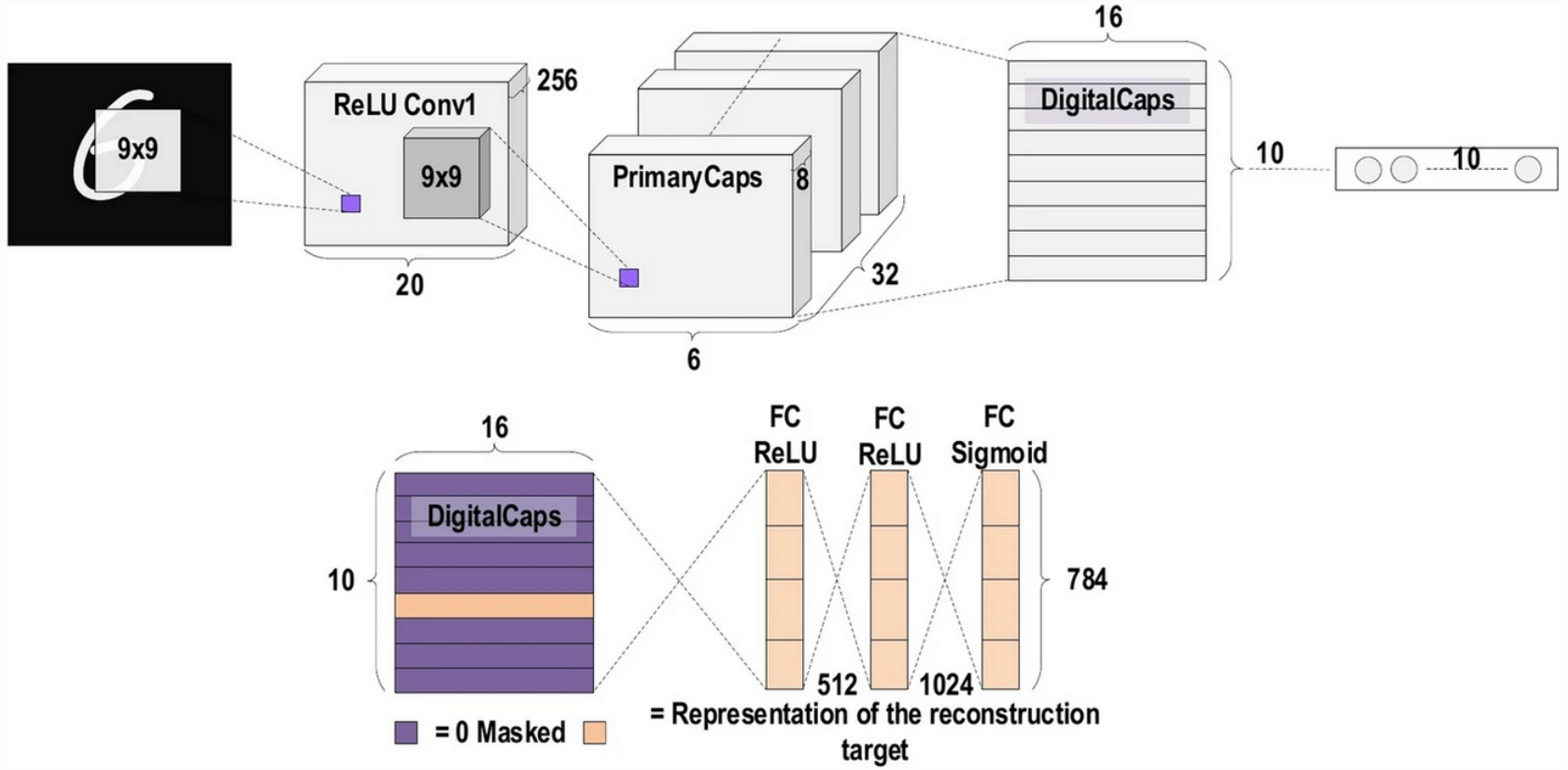
CapsNets

Dynamic routing involves lower capsules sending data to the most suitable parent capsule based on dot product.
The parent capsule is chosen through an agreement mechanism, utilizing the highest dot product between prediction vectors from lower capsules and the weight matrix.

Example. Suppose we have a picture of a house in different types of viewpoints. A CNN can recognize the front view of the house very easily since it is similar to the training set but it will have serious troubles in identifying the picture of the house from the top. Capsules on the other hand detect the roof and walls very easily and analyze the constant part in the image, i.e, the co-ordinate frame of the house capsule with respect to both roof and walls. The prediction is done by both the roof and the walls so as to decide whether the object is a house or not. These predictions are then sent to the mid-level capsule to check if the prediction of the roof and the walls matches each other..
This process is called **Routing by Agreement**.

CapsuleNet - Architecture

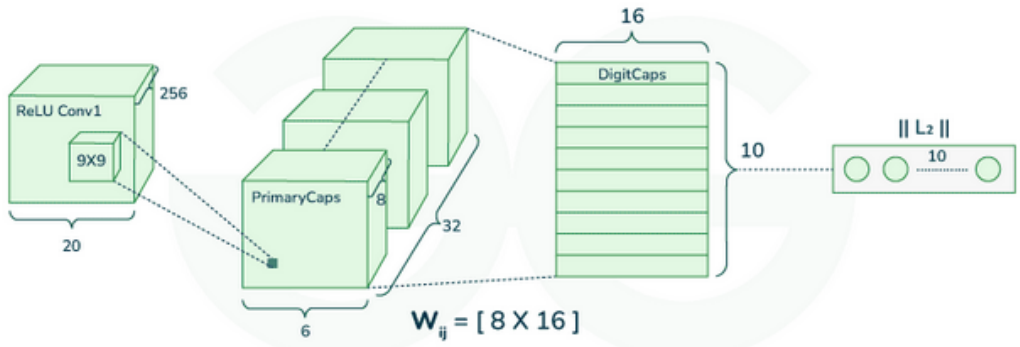
The CapsNet architecture consists of two main blocks, **Encoder** and **Decoder** networks, each one made up of various elements.



Encoding (top) and Decoder unit (bottom) networks

CapsuleNet - Architecture

The **Encoder** takes the image input and displays the image as a vector that contains all the instantiation parameters needed to render the image.



It contains

1. Convolutional layer to extract low-level input features.
2. PrimaryCaps layer uses capsules to record significant patterns. Every capsule in the input represents an instantiation parameter (like posture) of a certain object.
3. DigitCaps layer encode the instantiation parameters and likelihood that the object exists.

CapsuleNet - Architecture

The instantiation parameters are encoded using a **reconstruction loss**. Every training sample's loss is computed against every output class and all of the digit capsules' losses added together equals the total loss:

$$L_k = T_k \max(0, m^+ - \|v_k\|)^2 + \lambda(1 - T_k) \max(0, \|v_k\| - m^-)^2$$

where:

L_k is the margin loss for the k-th digit capsule.

T_k is a binary indicator.

v_k is the activity vector of the k-th digit capsule.

$\|v_k\|$ is length of the activity vector.

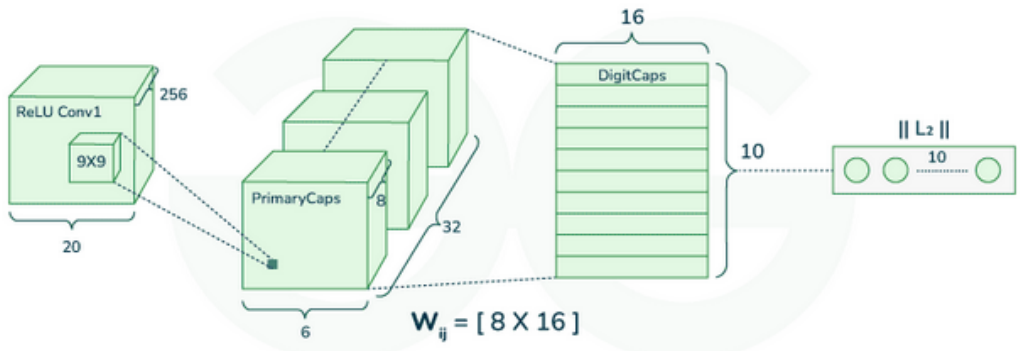
m^+ is the positive margin and m^- is the negative margin.

λ is a downweighting factor for the loss from incorrect digit capsules.

This loss function promotes the length of the correct digit capsule to be greater than m^+ and the length of the incorrect digit capsules to be less than $^m-$. The contribution of both accurate and inaccurate predictions to the total loss is balanced by the down-weighting factor.

CapsuleNet - Architecture

The **Decoder** recreates input images from the data contained in the DigitCapsules. The instantiation properties (such pose and viewpoint) of the DigitCapsules are utilized to rebuild the input data in the CapsNet following dynamic routing.



To facilitate faithful image reconstruction, the instantiation parameters are converted to the original input space. To promote accurate picture recovery, the loss for reconstruction is the Euclidean distance between the reconstructed image and the original input. Reconstruction improves classification accuracy and meaningful picture reconstruction, which furthers the overall training goal of Capsule Networks

CapsuleNets

More details on CapsuleNets:

Lecture by Sargur Srihari

Applications of Capsule Networks

- Self-supervised learning: *Canonical Capsules: Self-Supervised Capsules in Canonical Pose*
- Medical Image classification: *Fast CapsNet for Lung Cancer Screening*; *3D Res-CapsNet convolutional neural network on automated breast ultrasound tumor diagnosis*
- Anomaly Detection: *Abnormality detection in musculoskeletal radiographs using capsule network*.
- Autonomous Vehicles: *Traffic-light sign recognition using capsule network*

Region Based CNNs

The purpose of **Region Based CNNs (R-CNNs)** is to solve the problem of **object detection**.

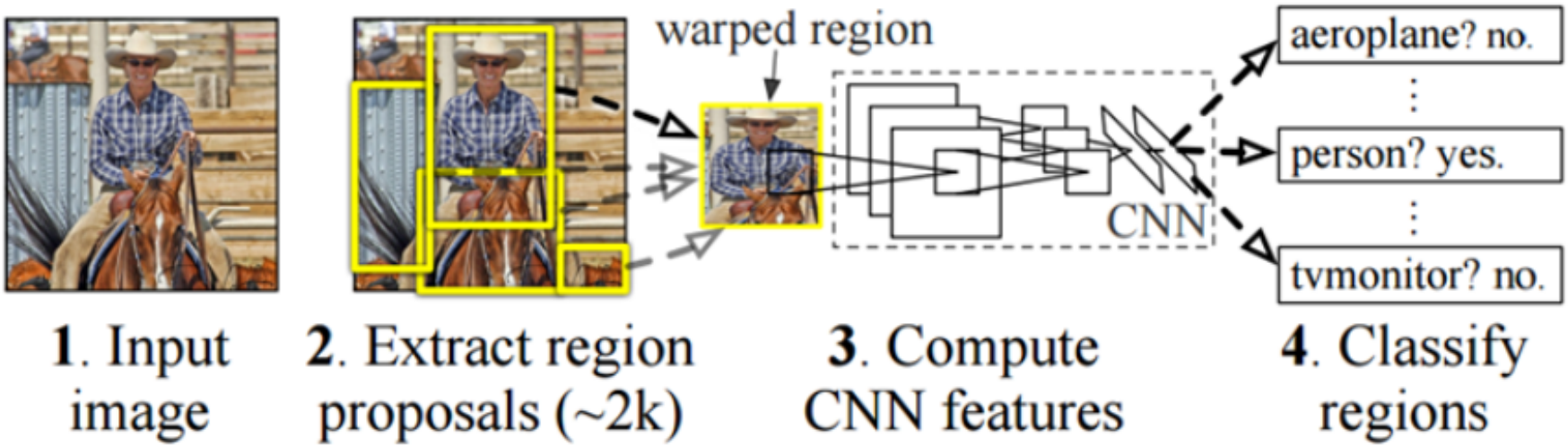
The advent of R-CNN has been more impactful than any of the previous papers on new network architectures [Girshick, Donahue, Darrell, Malik, 2014], with the original paper being cited over 35,000 times.

The original publications spurred one of the most dramatic advancements in computer vision and was followed by **Fast R-CNN** [Girshick, 2015] and **Faster R-CNN** [Ren, He, Girshick, Su, 2016] to make the model faster and better suited for modern object detection tasks

Region Based CNNs

Given a certain image, we want to **draw bounding boxes over all of the objects.**

The process can be split into two general components: (i) the region proposal step and (ii) the classification step.



Region Based CNNs

The authors note that any class agnostic region proposal method should fit. Selective Search is used in particular for RCNN. Selective Search performs the function of generating 2000 different regions that have the highest probability of containing an object. After we've come up with a set of region proposals, these proposals are then "warped" into an image size that can be fed into a trained CNN (AlexNet in this case) that extracts a feature vector for each region. This vector is then used as the input to a set of linear SVMs that are trained for each class and output a classification. The vector also gets fed into a bounding box regressor to obtain the most accurate coordinates.
As evident by their titles, Fast R-CNN and Faster R-CNN worked to make the model faster and better suited for modern object detection tasks.

Fast R-CNN

Fast R-CNN

Improvements were made to the original model because of 3 main problems. Training took multiple stages (ConvNets to SVMs to bounding box regressors), was computationally expensive, and was extremely slow (RCNN took 53 seconds per image). Fast R-CNN was able to solve the problem of speed by basically sharing computation of the conv layers between different proposals and swapping the order of generating region proposals and running the CNN. In this model, the image is first fed through a ConvNet, features of the region proposals are obtained from the last feature map of the ConvNet (check section 2.1 of the paper for more details), and lastly we have our fully connected layers as well as our regression and classification heads.

Faster R-CNN

Faster R-CNN works to combat the somewhat complex training pipeline that both R-CNN and Fast R-CNN exhibited. The authors insert a region proposal network (RPN) after the last convolutional layer. This network is able to just look at the last convolutional feature map and produce region proposals from that. From that stage, the same pipeline as R-CNN is used (ROI pooling, FC, and then classification and regression heads).
Being able to determine that a specific object is in an image is one thing, but being able to determine that object's exact location is a huge jump in knowledge for the computer. Faster R-CNN has become the standard for object detection programs today.