# Regularization in CNNs

Week 2: Computer Vision

## Agenda

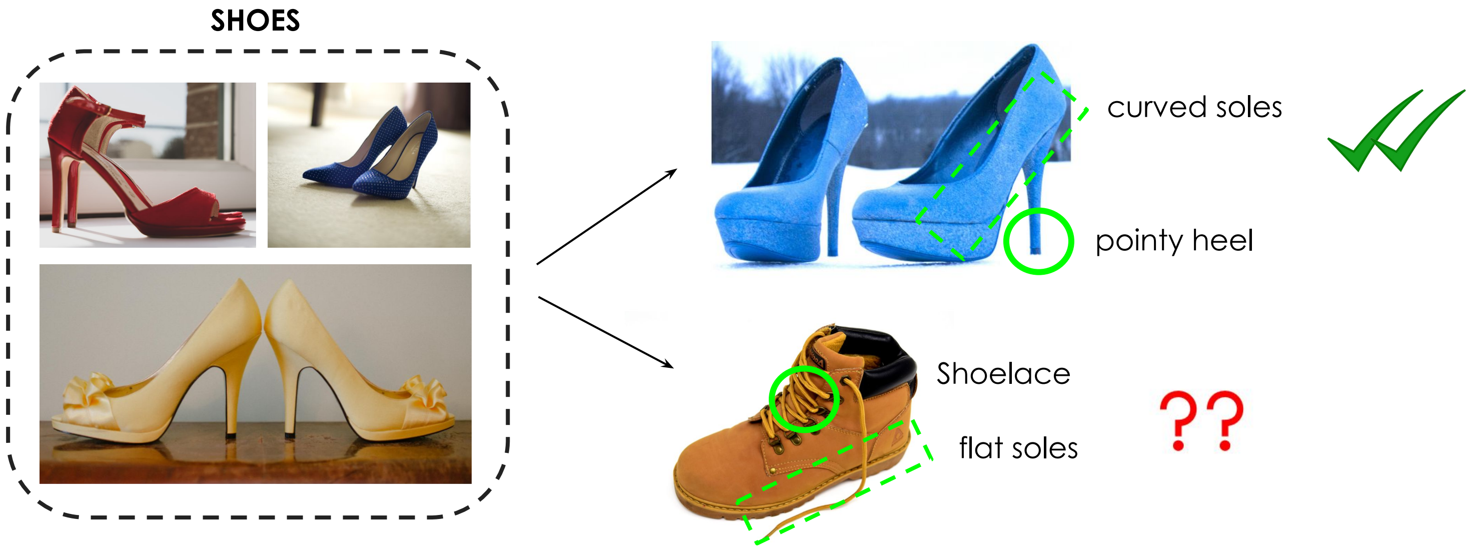
- The Overfitting problem for CNNs
- Techniques to address Overfitting
    - Data Augmentation
    - Batch Normalization
    - Spatial Dropout

## The Overfitting problem for CNNs

## Overfitting in CNNs

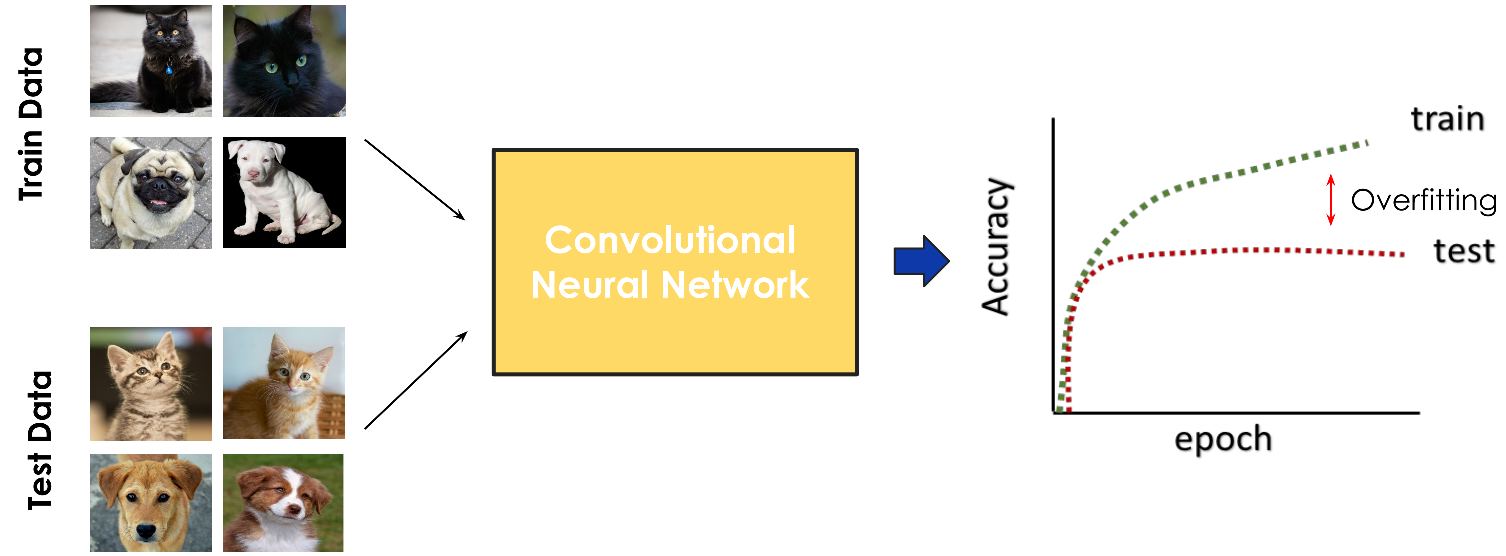
- If we only have a limited amount of data to train on, a neural network may be good at recognizing the data that it has been trained. **However, it may not perform well on new data, i.e., data it has never seen before.**

  The following example will help us understand this:



## Overfitting in CNN

- The same is true for Convolutional Neural Networks. They may not be able to perform well on the test dataset, if they have been overtrained on the training dataset. There are several approaches to potentially prevent this overfitting problem.



## Techniques to address Overfitting

## 1. Data Augmentation

- CNNs can learn filters that fire when a pattern is presented in a given orientation.

  However, individual filters in a CNN are **not invariant to the rotation of the image.**

  For example, If we train our model only with images like Image-1, the model might think that the pointy ears at the top of the images are a feature of a cat. So, when we test the same model with images like Image-2, which have a rotational component, it will look for pointy ears only at the top of the testing images and not find them, resulting in poor test performance.

- This can be avoided **if the training data itself includes rotated images.** But finding such data can often be time-consuming and expensive. That is where **Data Augmentation** proves its worth.

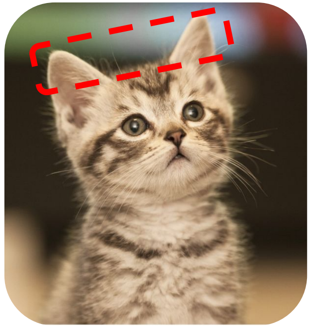


**Image-1**

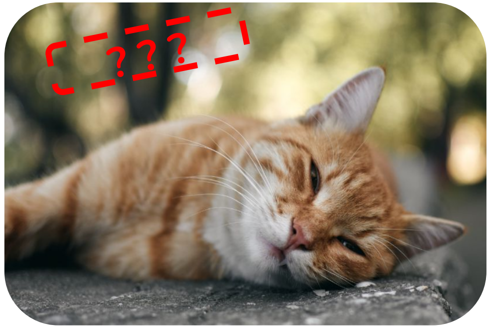


**Image-2**

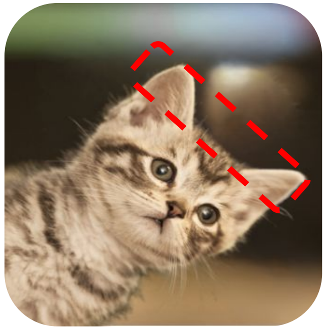


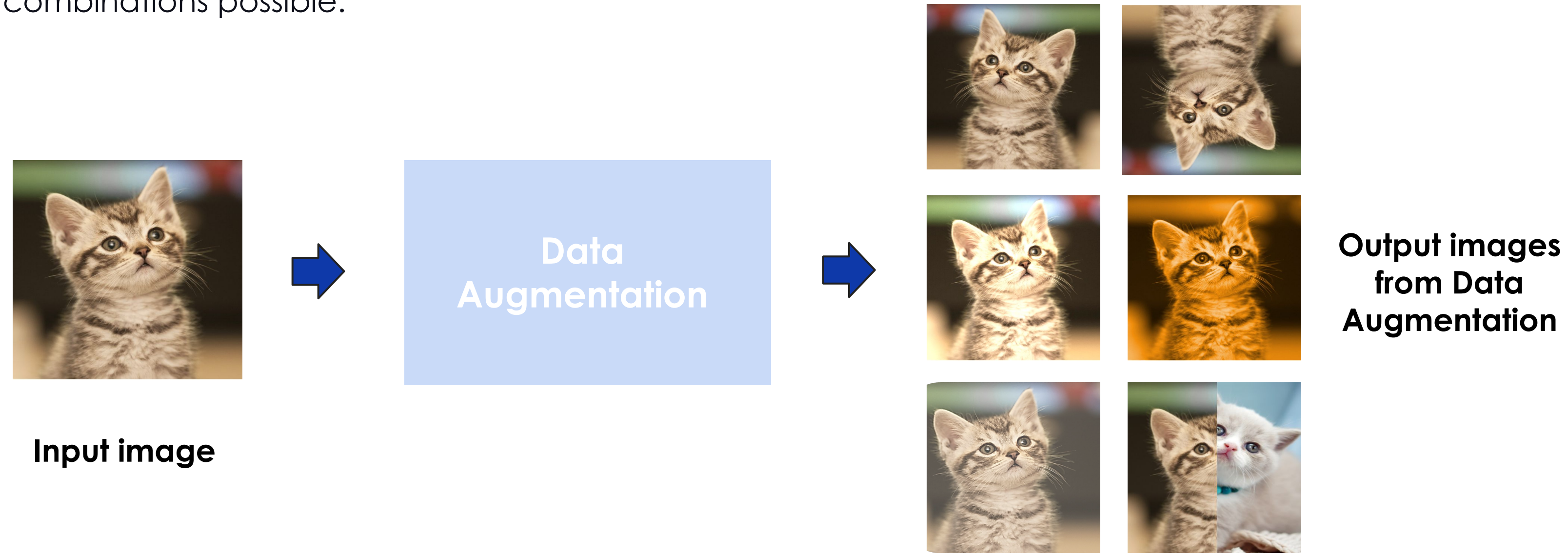
**Image-1 after Rotation**

## 1. Data Augmentation

- **Data Augmentation** is a technique for artificially increasing the size of a training set by generating data from the existing one. Some of the most popular image data augmentation techniques:
    - **Geometric transformations** — Images can be randomly flipped, cropped, rotated, or translated, and that's just the tip of the iceberg.
    - **Mixing images** — It is the process of combining different images. Two methods for combining photos are pixel averaging and crop overlaying.
    - **Color space transformations** — Changing brightness and contrast, changing RGB color channels, intensify any color.
    - **Random erasing** — Deletes a part of the initial image that forces the model to focus on the entire image rather than a portion of it. This method is inspired by the dropout method that zeros out a random fraction of weights in a neural network layer.
    - **Kernel filters** — This technique is used to sharpen or blur images using filters.

## 1. Data Augmentation

- The advantage of data augmentation is that **all the methods can be used simultaneously,** resulting in a large number of distinct data samples from the initial one. We could potentially create millions of images from a single image, due to the exponential growth in the number of combinations possible.



## 2. Batch Normalization

- Before starting with **Batch Normalization**, let's discuss why we need normalization and how it helps in speeding up the training process.

  The pixel values of an image represent a color code. If we use the images as they are and send them through a Deep Neural Network, the computation of these high numeric values may get costly. Normalizing the values to a range between 0 to 1 will reduce the computational expense.

  Consider the following example:

## 2. Batch Normalization

- Moreover, data points with high values in the non-normalized data set can induce **neural network instability.** This occurs due to the comparatively large inputs cascading down through the network's layers, which may lead to imbalanced gradients, resulting in the well-known **exploding gradient** problem.

**So how does normalization help?**

- Normalization reduces variation between the data points by placing all the data on the same scale. This helps prevent exploding gradient problems as there's less variation within data points. It turns out that scaling data points also makes training a network more efficient and faster.

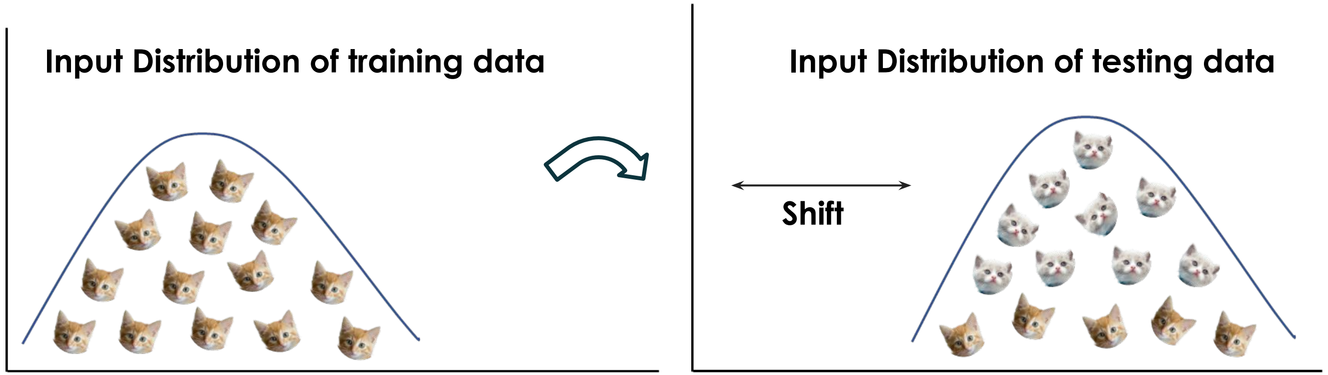
**How is Batch Normalization different from Normalization?**

- In deep learning, rather than just performing normalization once in the beginning, **Batch Normalization** is proposed as a technique to **scale the output of each layer**, specifically by standardizing the activations of each input variable per mini-batch.

## 2. Batch Normalization

- Batch Normalization speeds up the training of neural networks by reducing the **internal covariate shift**. To better understand covariate shift, consider the following example:

  Assume that for a cats and dogs classification problem, our model has only been trained on orange cats. Now, if we were to provide images of white cats while testing, the model may not perform effectively. This is because of the shift in the input distribution from orange to white, which is known as a **Covariate shift**.


Input Distribution of training data
Input Distribution of testing data
Shift

## 2. Batch Normalization

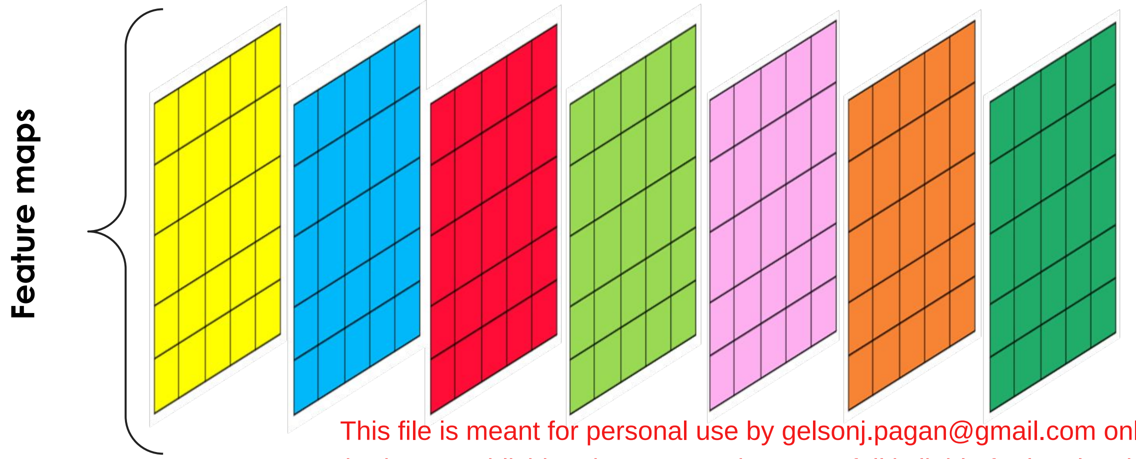
- The change in the distribution of inputs to different layers is referred to as the **internal covariate shift.** While training, each layer tries to fix the error that was introduced during forward propagation. Since every layer is attempting to fix the error that has been created, the update procedure is forever chasing a moving target. More specifically, this process forces the layers to learn from new 'input distributions' during an update.

### So how does Batch Normalization help?

- **Batch Normalization standardizes the activations of each input variable per mini-batch**, which means that during the weight update, the assumptions made by the subsequent layer regarding the spread and distribution of inputs will not alter drastically. It turns out that when the distribution of inputs to each layer is similar, training a network is more efficient and faster.
- In addition to speeding up learning, Batch Normalization (in short Batch Norm) also provides a **weak form of regularization**. Since Batch Norm is performed on mini-batches, **it adds noise to the data, which results in regularization.**

## 3. Spatial Dropout

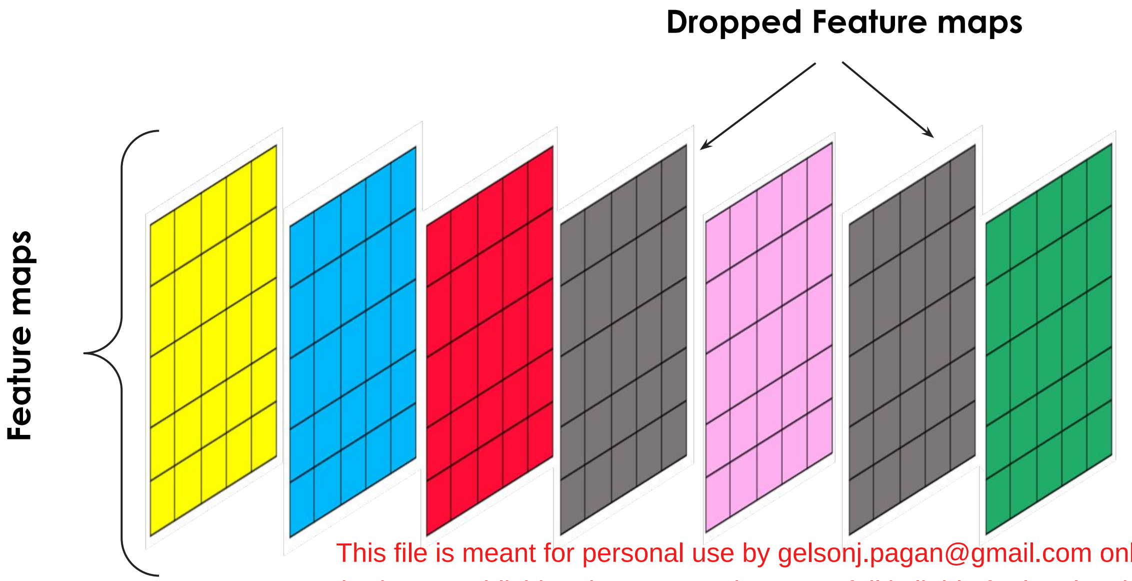
- We have already learned about the ‘**Dropout**’ regularization technique, which is used in neural network training to reduce overfitting. Let’s see how ‘Dropout’ is used after the convolution layer in CNNs.
- Convolution between a filter and an input produces feature maps in the convolution layer. In images, the values of neighboring pixels are highly correlated, resulting in correlated cells in feature maps. Thus, dropping the cells of feature maps at random using a basic dropout method may not actually prevent overfitting, because even if one cell is removed, the neighbouring cell could give a highly correlated gradient. That’s why we use another idea called **Spatial Dropout**.



## 3. Spatial Dropout

- In **Spatial Dropout**, whole feature maps themselves are randomly dropped. Dropping a feature map, means making all the cells of that feature map 0, which would be as good as not using it.

  Let's suppose we obtained 8 feature maps after the convolution operation. If spatial dropout is applied with dropout probability 0.25, then the expected number of dropped feature maps is 2.



## Summary

So to summarize, in this section we have learnt the following:

- What does overfitting mean with respect to convolutional neural networks, and what are the different techniques to address this issue?
- What is Data Augmentation, and what are some of the most popular DA methods?
- How do batch normalization and spatial dropout help in preventing overfitting?

This section covered the regularization techniques used to prevent overfitting in a CNN.

## Thank You