Draft 01 Summary of Art Classification Machine Learning Project

Ben Picker

Overview

The initial goal for our project is to be able to build a classifier. The classifier is an algorithm / formula that will take in an image and output its artistic style. Style will be determined from a set of fixed categories we decide on beforehand, using existing literature to inform us.

This project requires machine learning, specifically convolutional neural networks. Neural networks are a form of machine learning and, for image classification tasks, a particular type of neural networks called convolutional neural networks are the best tool.

This write up has two sections:

1. Part 1 explains the basic concepts behind Convolutional Neural Networks, which will be our primary tool during this project. It highlights their advantages and gives examples for how it can be applied.
2. Part 2 covers key papers existing in the literature on various papers. THIS SECTION IS STILL UNDER DEVELOPMENT AND UNFINISHED.

Part 1

Key Concepts Related to Convolutional Neural Networks

Basics of Machine Learning

What is machine learning?

Machine learning is a form of statistics. It originated from researchers who were trying to teach machines how to think. However, eventually there was a convergence between Artificial Intelligence researchers and statistics. Now, machine learning is at the intersection of several disciplines but the underlying logic uses the language of statistics.

Language of Machine Learning

All (supervised) machine learning methods have some common components:

- $\mathcal{X}$ an input data space
- $\mathcal{Y}$ an output space (categorical, numerical)
- A training data set $\{\mathbf{x}_i, y_i\}_{i=1}^n$

The goal of machine learning is to be able to take in any $\mathbf{x} \in \mathcal{X}$ predict $y \in \mathcal{Y}$.

Building the model

To do this, you have to do the following.

- choose a function class f (e.g. linear regression)
- optimize a set of parameters $\mathbf{w}$ using the training data so that $f(\mathbf{x}; \mathbf{w})$ predicts y. This is called a "model" (e.g. a linear regression model would be $f(\mathbf{x}; \mathbf{w}) = \mathbf{w} \cdot \mathbf{x} + \mathrm{b}$)
- test the model on a new data set to see if it predicts accurately.

Measuring Accuracy of Prediction

A machine learning model $f(\mathbf{x}; \mathbf{w})$ will make a prediction $\hat{y}$ for a given $\mathbf{x}$. Let y be the true value and $\hat{y}$ be the predicted value. A loss function is the error between our model's prediction and the actual expected output in the training set for a particular data point. It takes the form

$$\ell(y, \hat{y}) = \text{error}$$

where $\ell: \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$

However, the loss function by itself isn't what we minimize.

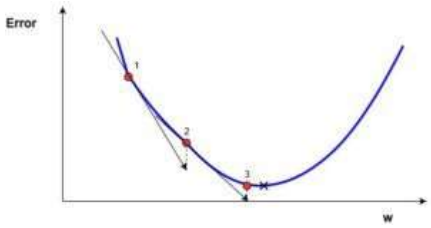
If we are given a data set, we can apply ℓ to each data point and compute the average loss per data point. This is called *empirical loss*

$$\mathcal{L}(\mathbf{w}, \mathbf{X}, \mathbf{y}) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(\mathbf{x}_i; \mathbf{w}), y_i)$$

This is what we minimize. To optimize our functions, we do the following:

- Define a loss function ℓ

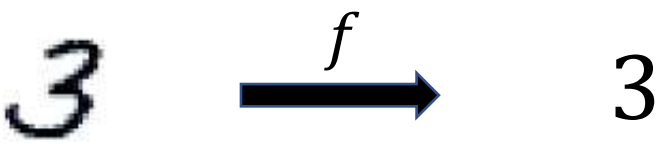
- Calculate the empirical loss L given $\mathbf{w}$
- Use a method called gradient decent to optimize $\mathbf{w}$ until the loss is low.



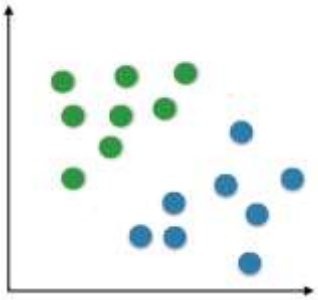
Classification:

A common machine learning task is *classification*. Classification maps inputs to a finite set of possible outcomes, called classes. For instance, a handwriting digit classifier would look like this

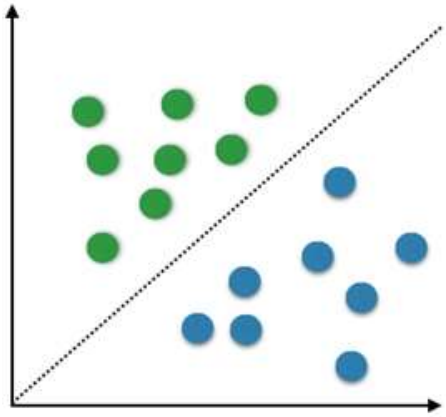
$$f(\mathbf{image}\,;\mathbf{w}) = \mathbf{digit}$$



Geometrically, the simplest type of classifier is a line. Suppose we wanted to classify a set of points.



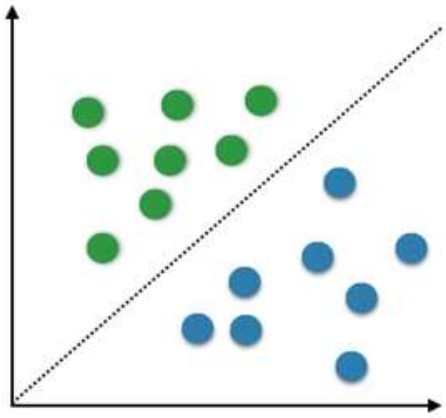
We could use a line to separate them. Our machine learning model would then take in these 16 data points $\{(x,y)\}_{i=1}^{16}$ and we would train a model to distinguish them.



$$\hat{y} = \text{sign}(b + \hat{\mathbf{w}} \cdot \mathbf{x})$$

The Need for More Sophisticated Classification

Our earlier example had only one classifier.



$$\hat{y} = \text{sign}(b + \hat{\mathbf{w}} \cdot \mathbf{x})$$

where $+1 = \text{green}$ and $-1 = \text{blue}$. But suppose you are trying to classify handwritten letters of the English alphabet. In this case, you would have 26 distinct classes.

$$\mathcal{Y} = \{\text{A}, \text{B}, \text{C}, \ldots, \text{Z}\}$$

Then you would need 26 classifiers:

$$\hat{y}_A = \text{sign}(b_A + \mathbf{w}_A \cdot \mathbf{x})$$
$$\hat{y}_B = \text{sign}(b_B + \mathbf{w}_A \cdot \mathbf{x})$$
$$\vdots$$
$$\hat{y}_Z = \text{sign}(b_Z + \mathbf{w}_Z \cdot \mathbf{x})$$

But we have a problem now. Because you cannot consider the classes in isolation from each other. When training our model (i.e. the collection of classifiers), we have factor in how they will affect each other.

Overview of Neural Networks

Neural networks are one method for doing multi-class classification.

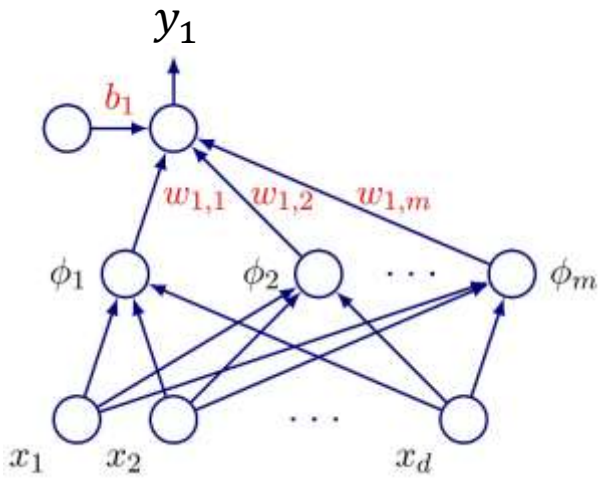
- $\mathbf{x} \in \mathbb{R}^d$ be an input vector.
- $\mathcal{Y} = \{y_1, \ldots y_k\}$ be a set of classes (e.g. $\mathcal{Y} = \{\text{A}, \text{B}, \text{C}, \ldots, \text{Z}\}$)
- $\boldsymbol{\phi}$: $\mathbb{R}^d \to \mathbb{R}^m$ some transformation
- $\mathbf{W}$ is $k \times m$ matrix, where each row represents the weights for class c
- $\mathbf{b} \in \mathbb{R}^k$

Consider a collection of functions:

$$\{f_c(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \mathbf{w}_c \cdot \boldsymbol{\phi}(\mathbf{x}) + b_c\}_{c=1}^k$$

where $\mathbf{w}_c$ is the cth row of $\mathbf{W}$ and b_c is the cth entry in $\mathbf{b}$.

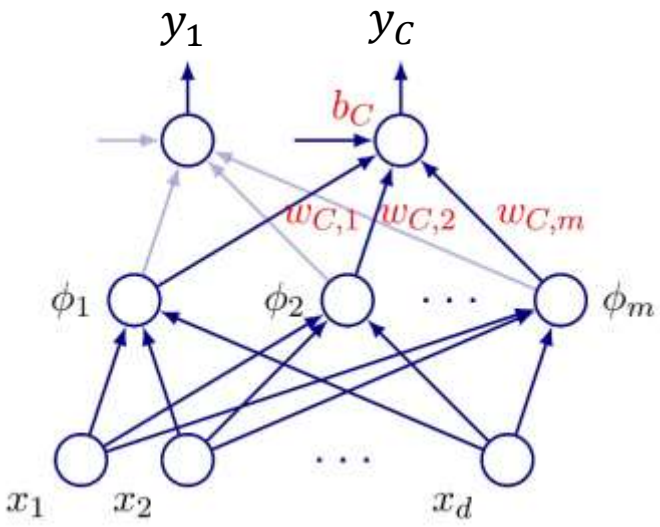
Visually, we can describe this equation using a graph with nodes and edges:



Notice that this is just a graphical representation of the mathematical function

$$f_1(\mathbf{x}; \mathbf{W}, \mathbf{b}) = \sigma(\mathbf{w}_1 \cdot \boldsymbol{\phi}(\mathbf{x}) + b_1)$$

However, if we have C classes, then this diagram needs more end point nodes and would look more like



Key point: these functions are essentially similar to the linear classifier from before. The only difference is now we have k functions instead of one. This is important because it means we are building on the ideas from the past, but now we have to take it a step further.

Shallow versus Deep Learning

Earlier in this write up, I showed how a line could be used to separate out two classes of points (blue points and green points). In the last section, I showed how you could then consider a collection of these functions, with one classifier for each function.

But why have only one layer of functions? What if we had 2 nested functions? Or 3? Or a bunch? This is called "deep learning", it means instead of just one function, we have several nested within each other.

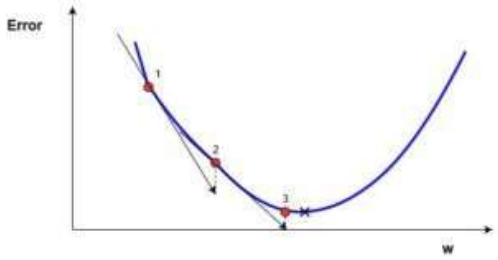
$$\{f_c(\mathbf{x}; \mathbf{W}_1, \mathbf{W}_2, \dots, \mathbf{W}_L) = F_L(F_{L-1}(\cdots F_2(F_1(\mathbf{x}; \mathbf{W}_1); \mathbf{W}_2)); \mathbf{W}_L)\}_{c=1}^{k}$$

Each F_L is a layer and has its own matrix $\mathbf{W}_L$ which has to be optimized as described at the beginning.

How Neural Networks Learn

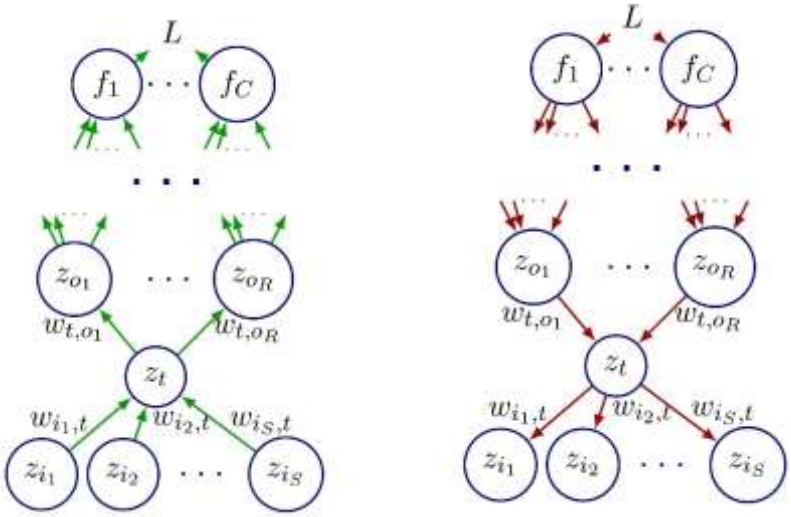
A neural network model is a collection of functions that we can optimize over its collection of weights $\{\mathbf{W}_i\}_{i=1}^{L}$ (L is the number of layers) to be a predictive model. This optimization is like any ML model in the sense we are just trying to take the parameters and optimize the model so it fits the data.

As mentioned previously, when we do optimization in machine learning, we often use gradient descent. Each iteration of the descent changes the parameters a bit and our model becomes more optimized to fit the data. Gradually, the loss L (i.e. the discrepancy between our predictions and the data) becomes smaller and smaller.



Neural networks however have a two-step process that allows the nodes to coordinate with each other.

FEEDFOWARD **BACKPROPOGATION**



So the discrepancy between our prediction and our error should be optimized over the whole network of parameters.

1. Compute L by computing all the layers.
2. Compute how L changes with respect to each for $\mathbf{W}_i$ and $\mathbf{b}_j$

These "changes" are derivatives. Therefore, we end up optimizing each parameter as follows

$$\left\{\mathbf{W}_i \coloneqq \mathbf{W}_i - \eta_i \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}\right\}_{i=1}^{L} \quad \left\{\mathbf{b}_j \coloneqq \mathbf{b}_j - \eta_j \frac{\partial \mathcal{L}}{\partial \mathbf{b}_j}\right\}_{j=1}^{L}$$

where $\mathcal{L}$ is the loss function, L is the number of layers in the network, and η_i is the learning rate (a scalar chosen by the designer) for $\mathbf{W}_i$ and η_j is the learning rate chosen for $\mathbf{b}_j$

Treating Images as Matrices for ML Functions

For our project, we want to be able to classify images by artistic style. To do this, we need to represent our images in a digital way. Images are represented as matrices.

From our discussions earlier, we can see that ML algorithms could of course accept a matrix as their input instead of just a vector. Let me explain that in a bit more detail.

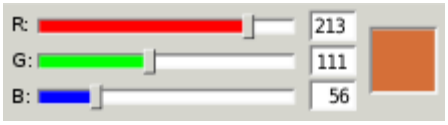
Images are represented as matrices

Images typically come in rectangles. They are comprised of pixels. On your computer monitor, a pixel is a color. But to a computer, a pixel is a number.

Our computers are able to convert numbers into colors because we have agreed upon systems for defining colors in terms of numbers.

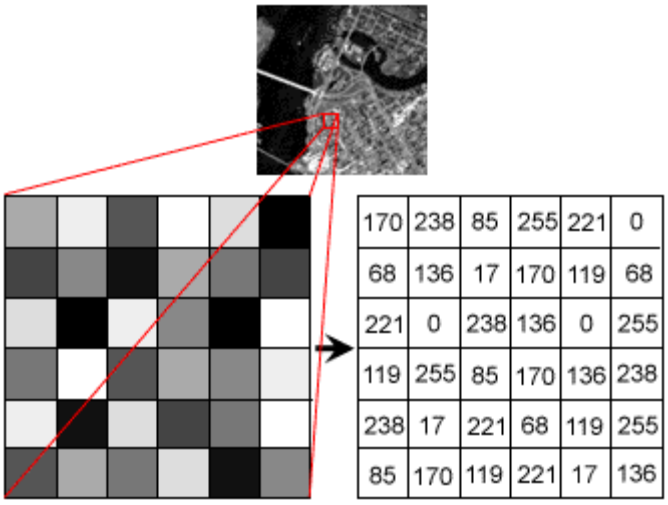
Most people know that you can mix colors to get other colors. The common system used by computers is the "Additive Color System" (RGB).

You can see here, by combining colors, new colors are made. On computers, this can often be done through an interface like this:



In this sense, every color a computer displays can be defined as a triple (r, g, b) where $r, g, b \in [0,255]$

Each pixel is then given this number and that's the color displayed. In the example below, you can see how the pixels for a patch of this photograph actually can easily be converted into a matrix.



Therefore, an image is a matrix where each entry is a triple (r, g, b) where $r, g, b \in [0,255]$. For example,

$$\begin{bmatrix} (50,160,220) & (255,0,0) \\ (112,48,160) & (125,210,130) \end{bmatrix}$$

would be an image like

Notice how the upper right-hand corner is pure red, corresponding to $(255,0,0)$ with no color in either G or B. Thus, we can mathematically define color in a way that we can apply mathematical functions to them as inputs. This is what machine learning does with images.

Convolutional Neural Networks (CNNs)

Why CNNs are needed

In the digit recognition case, an image comes in an our classifier outputs a digit.

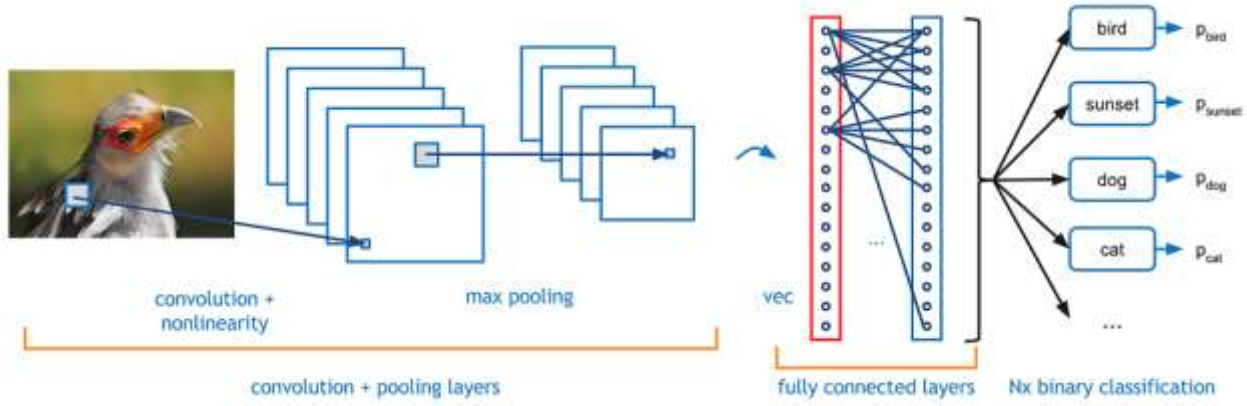
$$f(\textbf{image}\,;\textbf{w}) = \textbf{digit}$$

But digits are simple images. Numbers are essentially just lines and not complex, so simpler networks can be used. But what if you want to classify something more complex, like a type of animal. Then we need more advanced visual methods for classification.

Quick Look at CNNs Overall

CNNs have a basic structure to them.

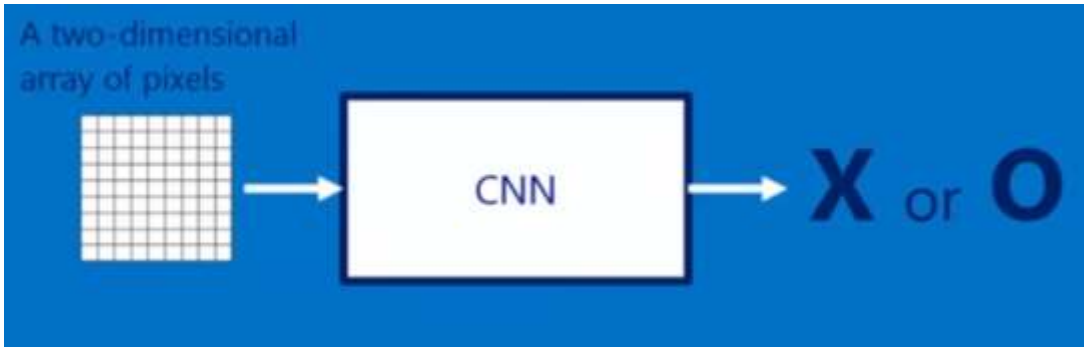
Overview of Process



Using CNNs to Classify Xs and Os

Suppose we have a two-dimensional array of pixels.

We will try to decide whether the pixels contain an X or an O.

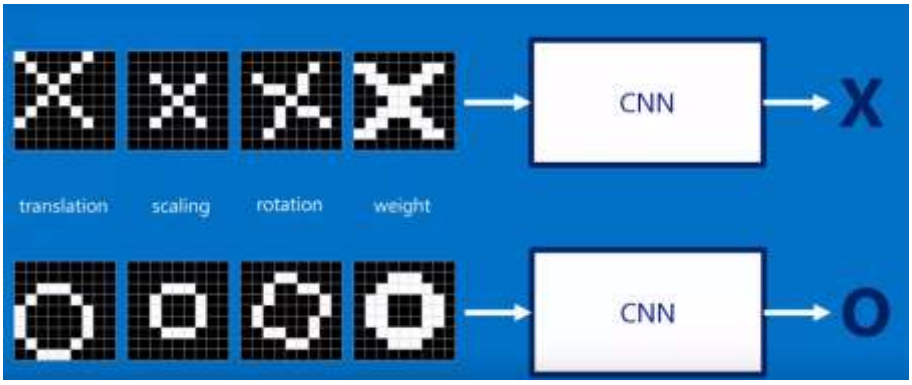


(source: https://www.youtube.com/watch?v=FmpDIaiMIeA)

Consider the following 9 × 9 grid.

-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	-1	-1	1	-1	-1	-1	-1
-1	-1	-1	1	-1	1	-1	-1	-1
-1	-1	1	-1	-1	-1	1	-1	-1
-1	1	-1	-1	-1	-1	-1	1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1

We want to classify this as an X. But shape classification isn't simple. An X can be distorted in a way that they look the same to a human but not to a computer. Here are some example distortions: Translation, Scaling, Rotation, Weight



(source: https://www.youtube.com/watch?v=FmpDIaiMIeA)

Convolution

In other words, even if there are discrepancies, the network must be capable of recognizing them as similar. A CNN uses feature layers to be able to recognize parts of an image. Suppose we had the following 3 features:

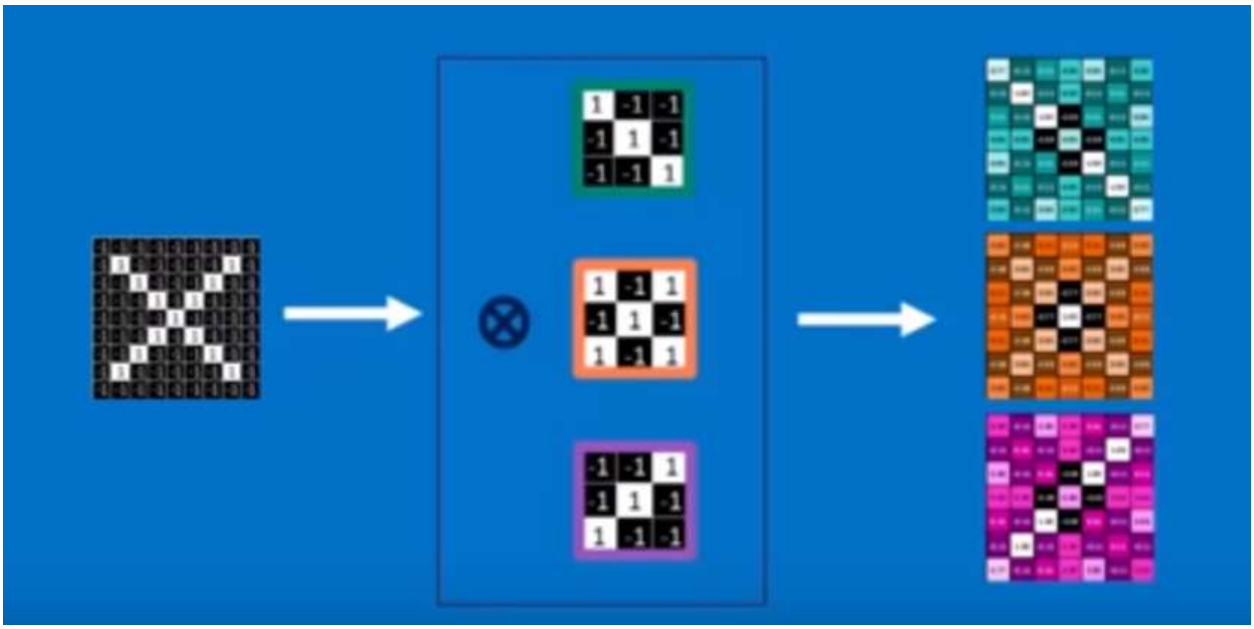
-1	-1	1
-1	1	-1
1	-1	-1

1	-1	1
-1	1	-1
1	-1	1

1	-1	-1
-1	1	-1
-1	-1	1

Note: The colors do not mean anything but are simply for illustration purposes to make the concepts easier to understand.

For a given feature, a layer of the CNN measures how well that particular feature matches every point in the grid 9×9 grid. Each layer represents a type of feature. Graphically, you can see this represented below.



(source: https://www.youtube.com/watch?v=FmpDIaiMIeA)

On the left, you have the 9×9 image. The box contains 3 possible features (marked with colors to help make it more obvious). And then this "layer" outputs the measurements for how well that feature matches particular areas of the image. This process of checking each area of the image to see the degree of matching with the feature is called "convolution."

Stages of CNNs

Remember, a deep learning model will in general mathematical terms look like this:

$$\mathbf{f}(\mathbf{image}; \mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_L) = F_L(F_{L-1}(\cdots F_2(F_1(\mathbf{image}; \mathbf{w}_1); \mathbf{w}_2)); \mathbf{w}_L)$$

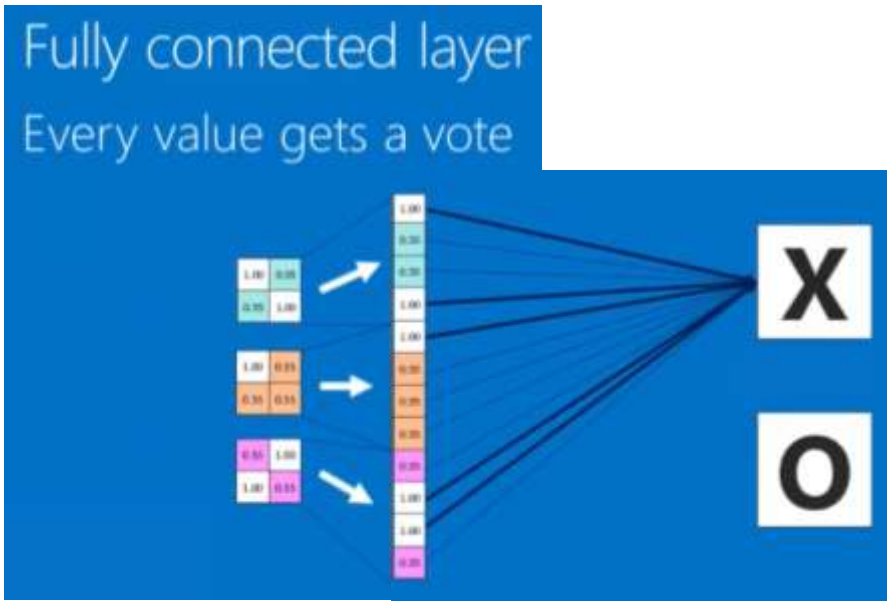
For convolutional neural networks, 3 common function classes that can be nested are

- Convolution
- ReLU, i.e. $\max(0, x)$
- Max Pooling

Visually, it looks like this:

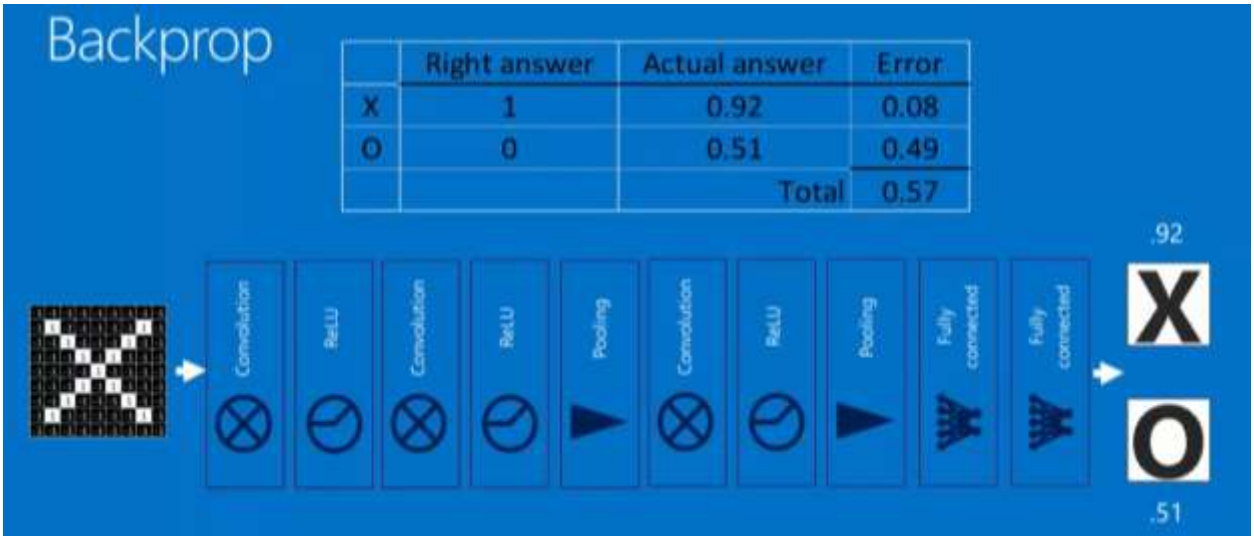
(source: https://www.youtube.com/watch?v=FmpDIaiMIeA)

The end product of nested convolutions, ReLU, and then get plugged into a neural network.



(source: https://www.youtube.com/watch?v=FmpDIaiMIeA)

This follows the same principles for neural networks described earlier in the document. The basic idea is, you tune the weights $\{\mathbf{W}_i\}_{i=1}^{L}$ for the nested algorithm. Then it is able to recognize features. The training can be imagined kind of like this

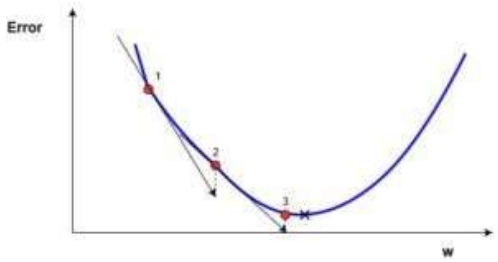


(source: https://www.youtube.com/watch?v=FmpDIaiMIeA)

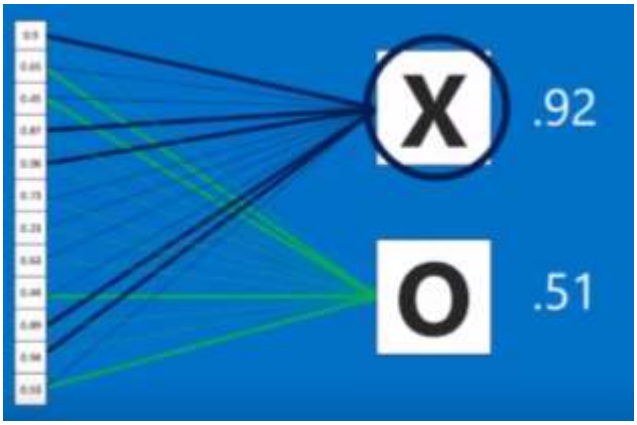
Again, there will be a collection of update functions

$$\left\{\mathbf{W}_i \coloneqq \mathbf{W}_i - \eta_i \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i}\right\}_{i=1}^{L} \quad \left\{\mathbf{b}_j \coloneqq \mathbf{b}_j - \eta_j \frac{\partial \mathcal{L}}{\partial \mathbf{b}_j}\right\}_{j=1}^{L}$$

And the goal is to find the lowest loss



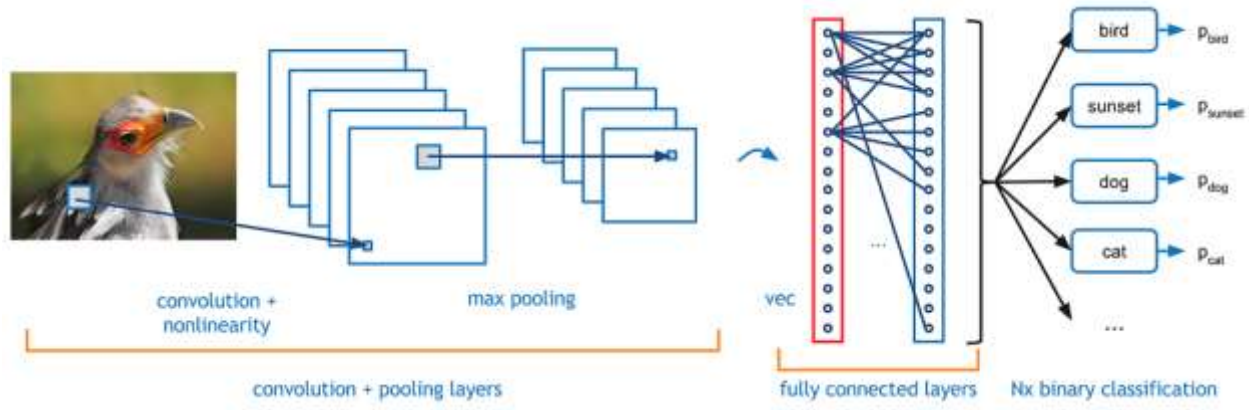
Then when you plug in a new image, you end up being able to classifying things. In this case, we expected an X and that's what we got.



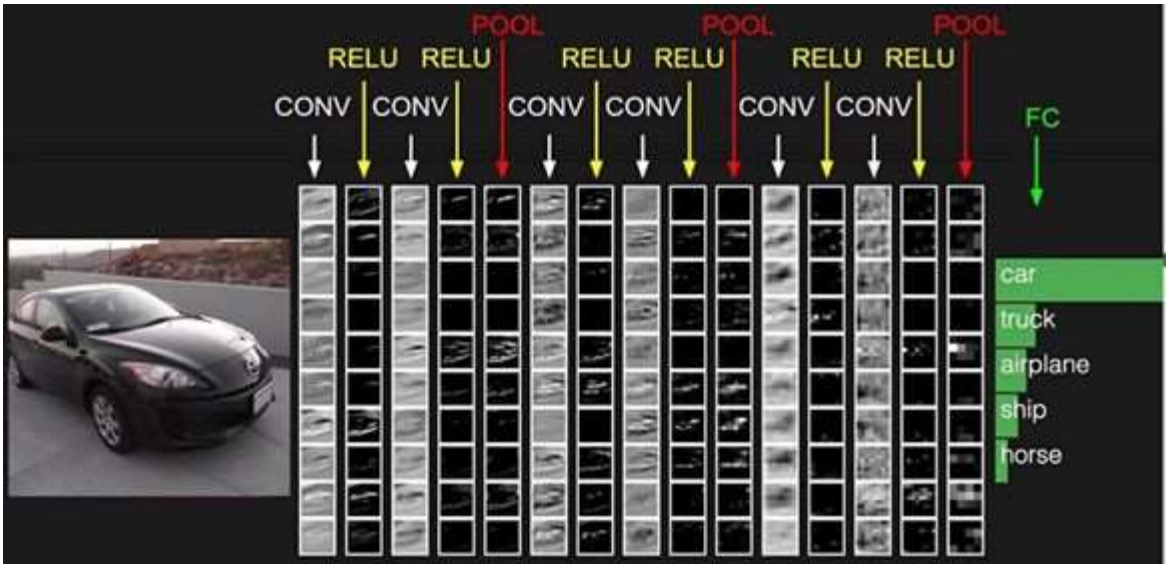
(source: https://www.youtube.com/watch?v=FmpDIaiMIeA)

CNN's Work on Complicated Images

The example with Xs and Os was to illustrate a simple example. Remember however, this isn't just applicable to X's and O's. We can use these to recognize information from much more complex images.



Here's an example with a car

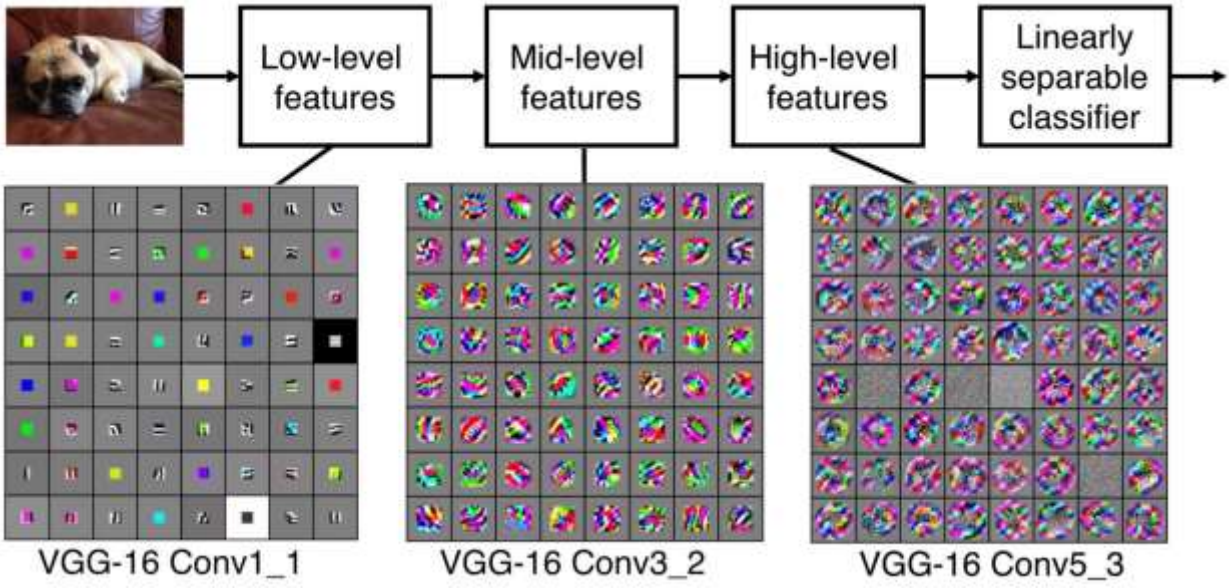


(http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf)

As you can see in this case, the final output wasn't something simple like an X or an O but it choose from categories of objects like the car.

You can also see here the "features", which are basically the intermediate attributes of algorithm. These often will seem unintelligible to a human. This is because some features are effectively "intermediate" stage features where they will look neither simple like a line nor complex like a car. These intermediate features may not be recognizable to us. They are tools the algorithm uses however and knows their meaning.

Here's another example



(http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture5.pdf)

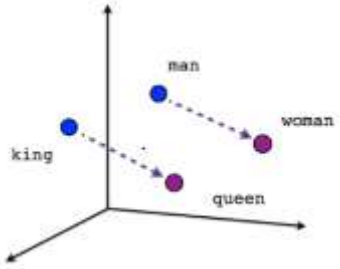
Mixing Image Analysis and Text Analysis ML Tools

Image Analysis

The above sections discussed how CNNs can create features.

Text Analysis

Recall, packages like word2vec represent the relationship between words in a corpus using vectors. These vectors can then be manipulated to explore and isolate semantic meaning. For example, consider the words "queen", "king", "woman, "man". If these words were represented as points in a vector space, simplistically it would look like this:



To find v_{king} we can do the following operation then:

$$v_{queen} - v_{woman} + v_{man} = v_{king}$$

Mixing the two: "Image Captioning"

Image captioning is the process of taking and image and outputting a sentence that matches what's happening in the

https://towardsdatascience.com/tag2image-and-image2tag-joint-representations-for-images-and-text-9ad4e5d0d99

MS COCO image n.248194

MS COCO captions

A man reading a paper and two people talking to a officer.

A man in a yellow jacket is looking at his phone with three others are in the background.

A police officer talking to people on a street.

A city street where a police officer and several people are standing.

A police officer who is riding a two wheeled motorized device.

As you can see in this example, the algorithm is capable of interpreting a variety of different aspects of the image. This is valuable because it allows us to start having machines characterize images with words.

This interplay between words and images will be crucial to keep in mind as we consider how we characterize art.

Part 2

Brief Literature Review

For the purposes of our work, we must keep in mind the different between supervised and unsupervised learning because they require substantially different data sets. If we wish to classify art by genre using supervised learning, the initial data set must already have the images classified. For example,

Cubism

Post-Impressionism

⋮ ⋮

Impressionism

Such a data set must have at a bare minimum,

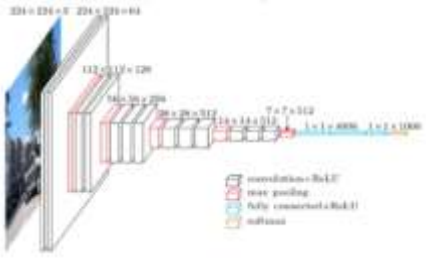
- Image files
- Style labels

<u>Notes from TTIC 31230:</u>

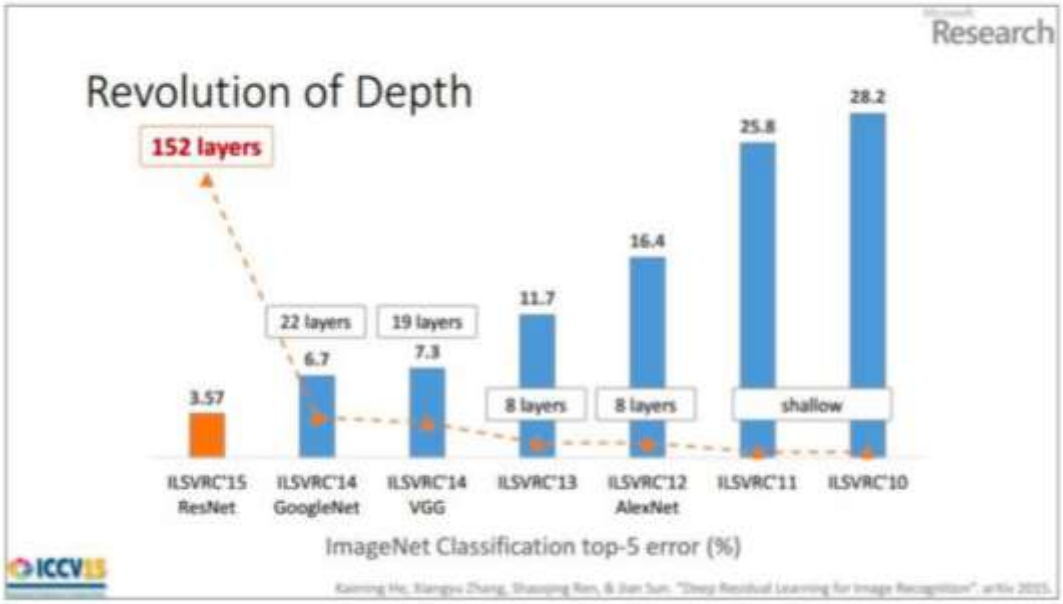
VGG

- ImageNet is a huge database of images for academic researchers. Researchers work with this data set to improve image recognition aclogrithm.
- VGG is a CNN architecture proposed in 2014. Performed exceptionally well in the 2014 Image Net Large Scale Visual Recognition Challenge (ILSVRC).
- Some fame is because not only does it work very well but the methods were made available for free online.

VGG



Here’s a comparison of some of the algorithms:



<u>This is important because it means the ML field changes quickly. So papers from prior to 2014 may be well out of date.</u>

Title: Historical Context-based Style Classification of Painting Images via Label Distribution Learning

Authors: Jufeng Yang, Liyi Chen, Le Zhang, Xiaoxiao Sun, Dongyu She, Shao-Ping Lu, Ming-Ming Cheng

Year: Octobeer 2018

- Method: Using CNNs combined with a hierarchical feature learning

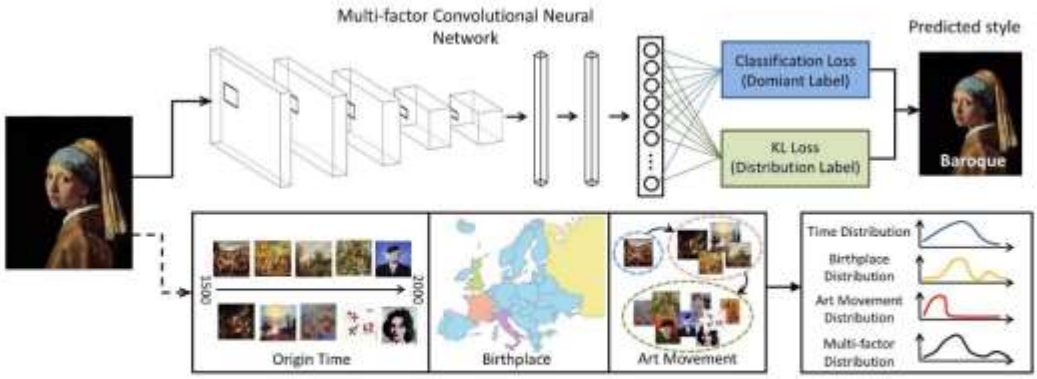


Figure 2: The illustration of the proposed method. Taking into account the three factors in the historical context that describe the relationship between styles (Origin Time, Birthplace, Art Movement), the framework simultaneously optimizes the classification loss and distribution loss. The softmax loss is employed as the classification loss, while the style distribution loss (KL loss) is used as an auxiliary task to assist visual feature learning towards better generalization ability.

Title: Painting-91: a large scale database for computational painting categorization

Authors: Fahad Shahbaz Kahn, Shida Beigpour, Joost van de Weijer, Michael Felsberg

Year: 2014

- Kahn et. al Comments on this field
  - Two common tasks researchers in art style identification are focused on:
    - Automatically recognizing the style of an image
    - Automatically recognizing the artist of an image
  - Substantial recent progress (as of 2014) on object and scene recognition
    - Local visual features via "bag of words" approach, similar to word2vec
  - As of 2014, there aren't any large data sets for classifying digital paintings (Kahn et. al, 2). Most image classification successful tasks have used large data sets.
- Papers Mentioned
  - Sablatnig explore classifying images brushstroke techniques http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.172.6599&rep=rep1&type=pdf
  - Shamir and Tarakhovsky also try to classify paintings by artistic movements https://dl.acm.org/citation.cfm?id=2307726
- Data Sets Mentioned

- [1]PrintART: “The artistic dataset consists of 988 monochromatic images. The dataset contains 27 theme classes. Unlike other datasets used for artist and style classification, this dataset contains local and global annotations for visual themes. Furthermore, pose annotations of images are also provided.”
- [2]Artistic genre data set: “This dataset comprises of 353 different paintings categorized into 5 artistic genres namely: abstract impressionism, cubism, impressionism, pop art and realism. The images have been collected from the internet”
- [3]Artistic style data set: “The artistic style dataset consists of 513 paintings of nine different painters. The artists included in the study are as follows: Monet, Pollock, Van Gogh, Kadnisky, Dali, Ernst and deChirico.”
- [4]Painting genre dataset: “The painting genre dataset is challenging and consists of 498 painting images. The styles used in their study are as follows: abstract expressionist, baroque, cubist, graffiti, impressionist and renaissance. The images have been collected from the internet.”
-

- Kahn et. al Data set:
  - created a data set of 4,266 images from 91 different painters and labeled each by style and artist.
  - They divide the data into a 2275 training set and a 1991 validation set.

Conclusion:

- Pros about paper
  - Possible data sets
  - Valuable summary of work in the field
- Cons about paper
  - Don’t use CNNs
  - Pre 2014

[1] Carneiro, G., da Silva, N.P., Bue, A.D., Costeira, J.P.: Artistic image classification: an analysis on the printart database. In: Proceedings of the ECCV (2012)
[2] Zujovic, J., Gandy, L., Friedman, S., Pardo, B., Pappas, T.: Classifying paintings by artistic genre: an analysis of features and classifiers. In: Proceedings of the MMSP (2009)
[3] Shamir, L., Macura, T., Orlov, N., Eckley, D.M., Goldberg, I.G.: Impressionism, expressionism, surrealism: automated recognition of painters and schools of art. ACM Trans. Appl. Percept. 7(2), 8–16 (2010)
[4] Siddiquie, B., Vitaladevuni, S.N.P., Davis, L.S.: Combining multiple kernels for efficient image classification. In: Proceedings of the WACV (2009)