D001 Economic Analysis of Non-Standard Data

Benjamin W. Arold

10. Images as Data

Outline

Human vision is interesting (not only for social scientists)

- Vision is an incredibly powerful sense
- Allows us to interact with the physical world without making any physical contact
- A substantial part of our brain is somehow involved in visual perception
- Human vision is interesting on many levels:
  - **Fundamental:** How do biological vision systems operate?
    $\rightarrow$ Not part of this class (or marginally)
  - **Aggregate/societal:** In what ways does what and how we see shape societal outcomes?
    $\rightarrow$ The focus of this class

Limitations of human vision

Figure: Football fans (DALL-E 2024 generated)

- Low quantification skills
- Speed of processing, and scalability
- Inconsistency and bias
- Environmental conditions, invisible spectrums
- One Solution: Computer Vision

Computer vision

- David Marr: Automating human visual processes
- Definition: Computer vision is the computational process of interpreting and understanding visual data to derive information about the structure and properties of the external world
- In this and next lecture: Look at techniques to analyze visual data in large quantities to study economics questions
  - We start where the visual data already exists
  - We do not cover, from a technical viewpoint, how to get that visual data (e.g., optics)

Studying images in economics and social sciences is not new

- Research has shown that images are key to agenda-setting and framing in newspaper coverage
- Images can impact people's perceptions of political candidates and their votes
- Encourage (or discourage) political participation
- Images seem to have a greater effect than written and spoken content in capturing attention
  - They ease information processing, improve recall, and evoke emotions[1].
- Distinction: Images as measures for latent concepts vs. images as own object of interest

[1] Examples of relevant papers can be found in Webb Williams, N., Casas, A., and Wilkerson, J. (2020). Images as Data for Social Science Research: An Introduction to Convolutional Neural Nets for Image Classification (Elements in Quantitative and Computational Methods for the Social Sciences). Cambridge: Cambridge University Press. doi:10.1017/9781108860741

Can we detect election fraud with images? (Cantu, 2019)

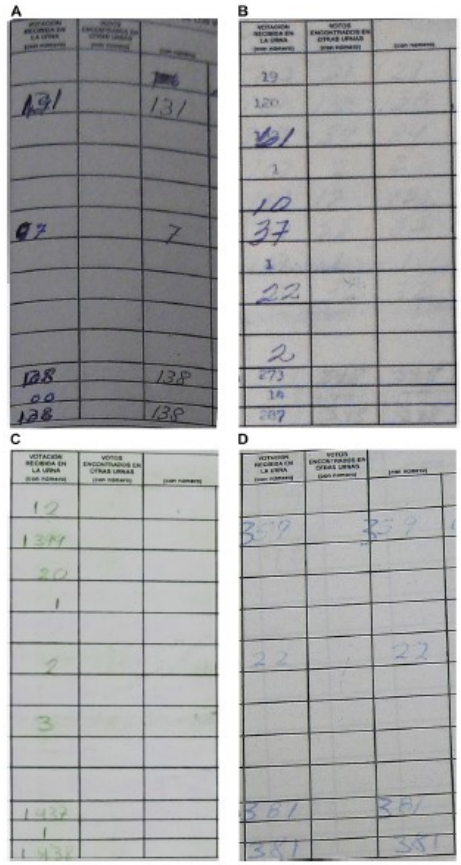


Figure: Examples of altered vote tallies in Mexico, 1988

Politicians' Instagram: what is popular? (Peng, 2021)

Figure: Screenshot of New Jersey Senator Bob Menendez' Instagram (2017)

Computer vision is increasingly important in social science

- Images significantly contribute to and mirror the social/political landscape
- Digitization has led to an increased accessibility of such images in everyday life
  - Consequently, this provides new and valuable opportunities for researchers and practitioners alike
- Advances in computer vision significantly ease the process of using images as data
  - For example, deep learning techniques for identifying objects, measuring visual sentiment, or recognizing faces
  - Innovation in computer vision has large societal effects

Digitization has led to an explosion in visual material

- There has been an explosion in the production and consumption of images
- On YouTube, an estimated 3.7 million videos are uploaded per day[2]
- Users upload 1.3 billion images to Instagram per day[3]
- Such figures are estimates and can change fast but are enormous in any case

[2] Adavelli, M., & Tonogbanua, L. (2023). How Many Videos Are Uploaded to YouTube A Day? https://techjury.net/blog/how-many-videos-are-uploaded-to-youtube-a-day/

[3] Growcoot, M. (2023). Almost All Photos Are Now Taken on Smartphones, According to Study. PetaPixel. https://petapixel.com/2023/06/20/almost-all-photos-are-now-taken-on-smartphones-according-to-study/

What is an image?

- Simplest form (single-channel): two-dimensional function $f(x, y)$.
  - Maps a coordinate pair to an integer/real value related to intensity/color.
  - Single-channel: Binary or monochrome, grayscale, black/white images
- Each point is called a pixel (picture element) or pel
- Multiple channels possible, such as RGB:
  - Color represented using three channels.
  - Pixel at point (x, y) represented as $(r_{x,y}, g_{x,y}, b_{x,y})$.
- Channel-specific pixel value represented as an integer between 0 and 255 or a floating-point value in [0, 1]

Array example

```
array([[131, 131, 131, ..., 107, 106, 106],
       [131, 132, 132, ..., 107, 107, 106],
       [132, 132, 133, ..., 108, 107, 106],
       ...,
       [192, 192, 192, ..., 189, 187, 185],
       [193, 193, 193, ..., 189, 187, 185],
       [194, 194, 194, ..., 188, 186, 184]], dtype=uint8)
```

Figure: An array of (549, 976)

What the array represents:

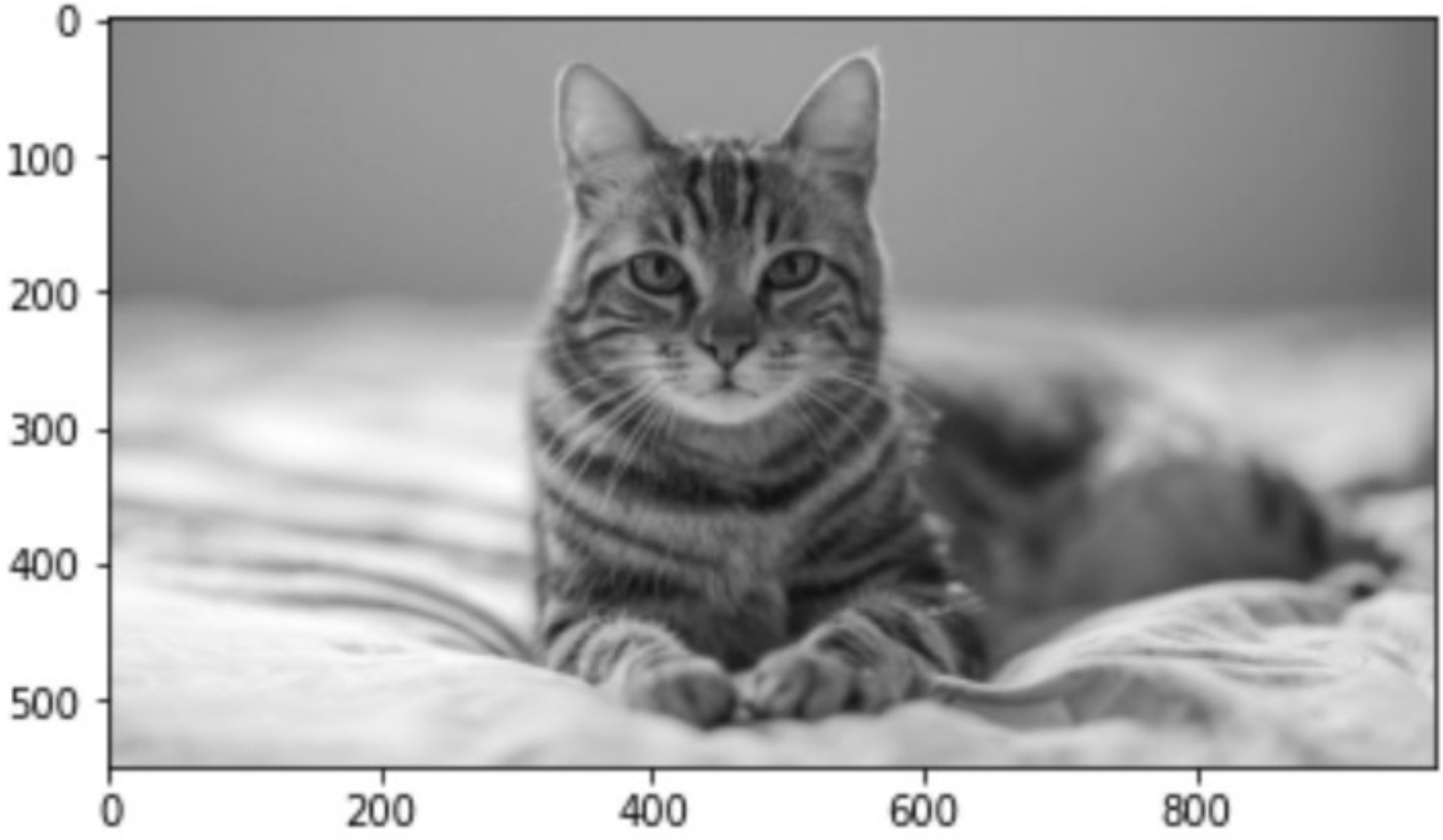


Figure: Meow

CNNs are the current computer vision workhorse

- Convolutional Neural Networks (CNNs or ConvNets) are specialized neural networks for CV tasks
  - For tasks such as object detection, facial recognition, image segmentation, sentiment analysis
- Much of the recent image-based social science research uses CNNs
- Other approaches include:
  - **Classic techniques:** Pre-deep learning relied on handcrafted features like color histograms, image entropy
  - **Other neural nets:** Autotaggers for skipping model training, GANs for image generation, RNNs for audio analysis, vision transformers for obscured image classification (among others)
  - **Hybrid approaches:** Combining traditional techniques with deep learning.

Outline

Classic techniques typically require explicit "rules"

- Imagine you search for images with a specific political leader
- Should you create a checklist of physical characteristics?
  - Color of their eyes, size of forehead?
  - Not robust (facing the camera? sunglasses?)
  - An infinite list of rules?
- Fairly general features can be helpful for various tasks
  - Corners
  - Blobs (dark/bright region)
- **Feature detection**: identification of points of interest
- Explicit feature detection in classic or "traditional" machine learning

Classic techniques: Graphical Illustration

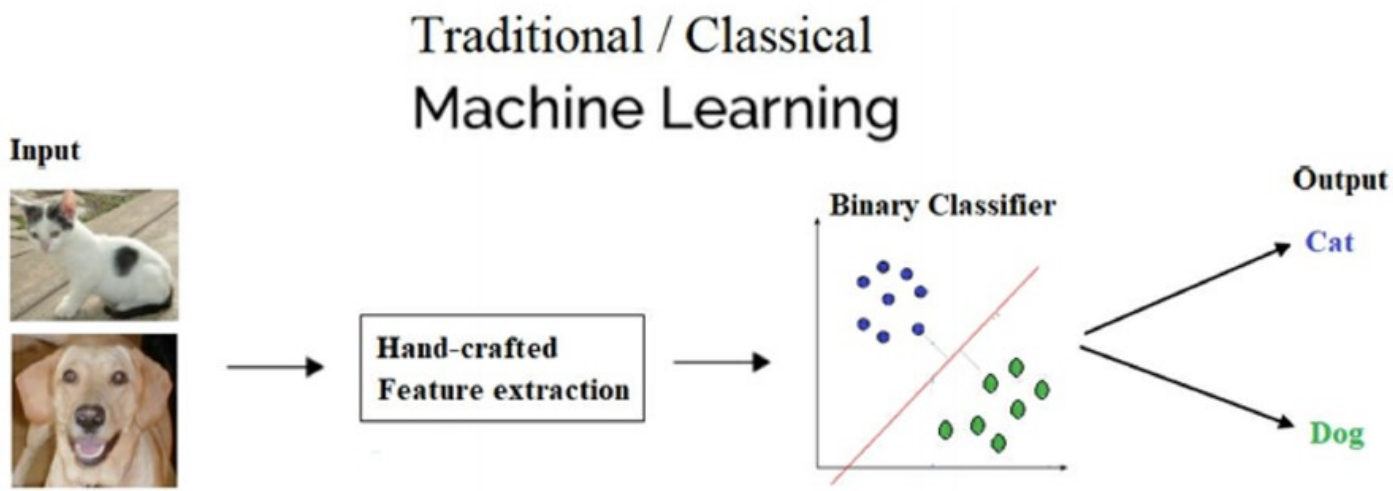


Figure: Dey (2018)[4]

[4] Dey, S. (2018). Hands-On Image Processing with Python: Expert Techniques for Advanced Image analysis and Effective Interpretation of Image Data. Packt Publishing Ltd.

Feature extraction: Examples

- **Color Histograms**: Represents the distribution of color values in an image
- **Texture Features**: Calculate texture features like entropy, energy, contrast, and homogeneity from the grayscale co-occurrence matrix
- **Haar-like Features**: Use integral images to efficiently compute features that recognize edges, lines, and rectangles for object detection

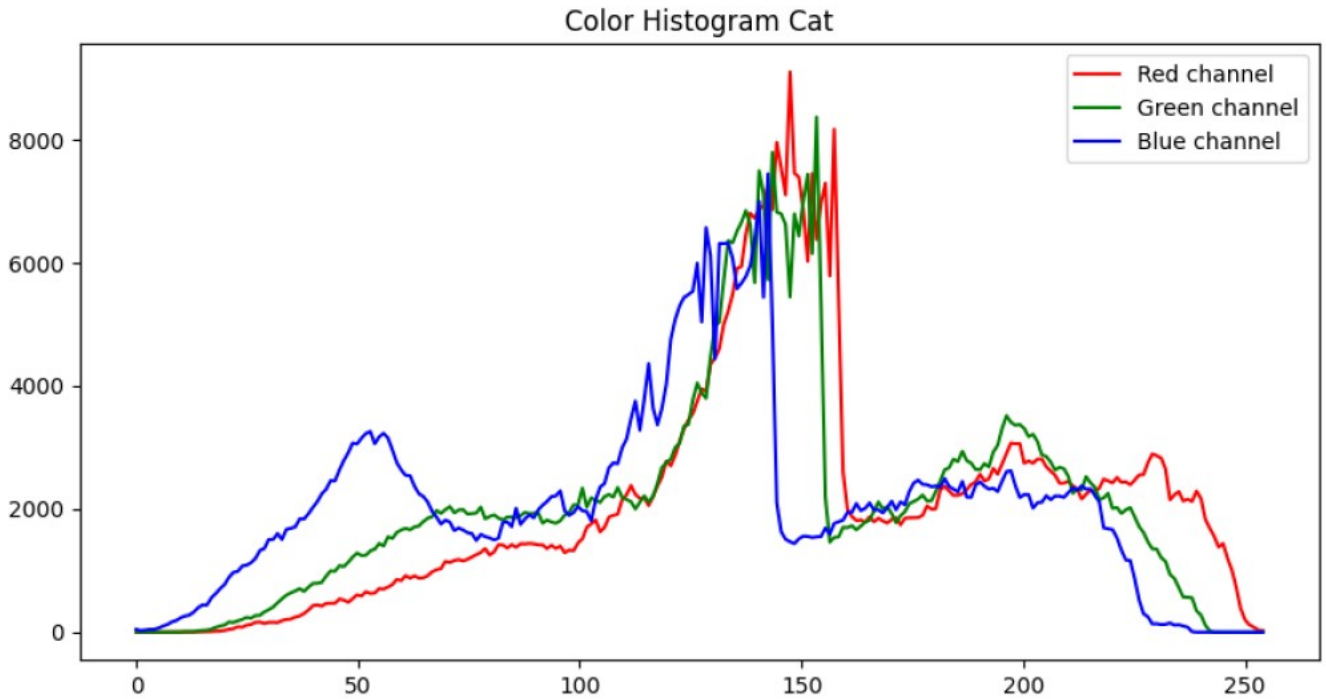
$\rightarrow$ Different feature types provide a different kind of information about the image content, and the best one to use depends on the specific task at hand

Color histograms show the pixel intensity distribution

- Represents the distribution of pixel intensity levels
- Mathematically, $h(r_k) = n_k$ where h is the histogram, r_k is the k-th intensity value, and n_k is the number of pixels with intensity r_k
- Typical use cases: image retrieval, color-based segmentation

What color histogram do you expect for the cat?

This cat's histogram shows the frequencies for each channel (RGB)



Texture features reveal image complexity and uniformity

- Texture features are typically based on the grayscale co-occurrence matrix P
- This matrix contains the occurrences of pairs of grayscale values at a given offset (distance and direction)
- Describes how often pairs of pixel intensity values (e.g., grayscale levels) occur in a specified spatial relationship within the image
- Measures the frequency of co-occurrence of pixel intensities at a specific offset (distance and direction) from each other

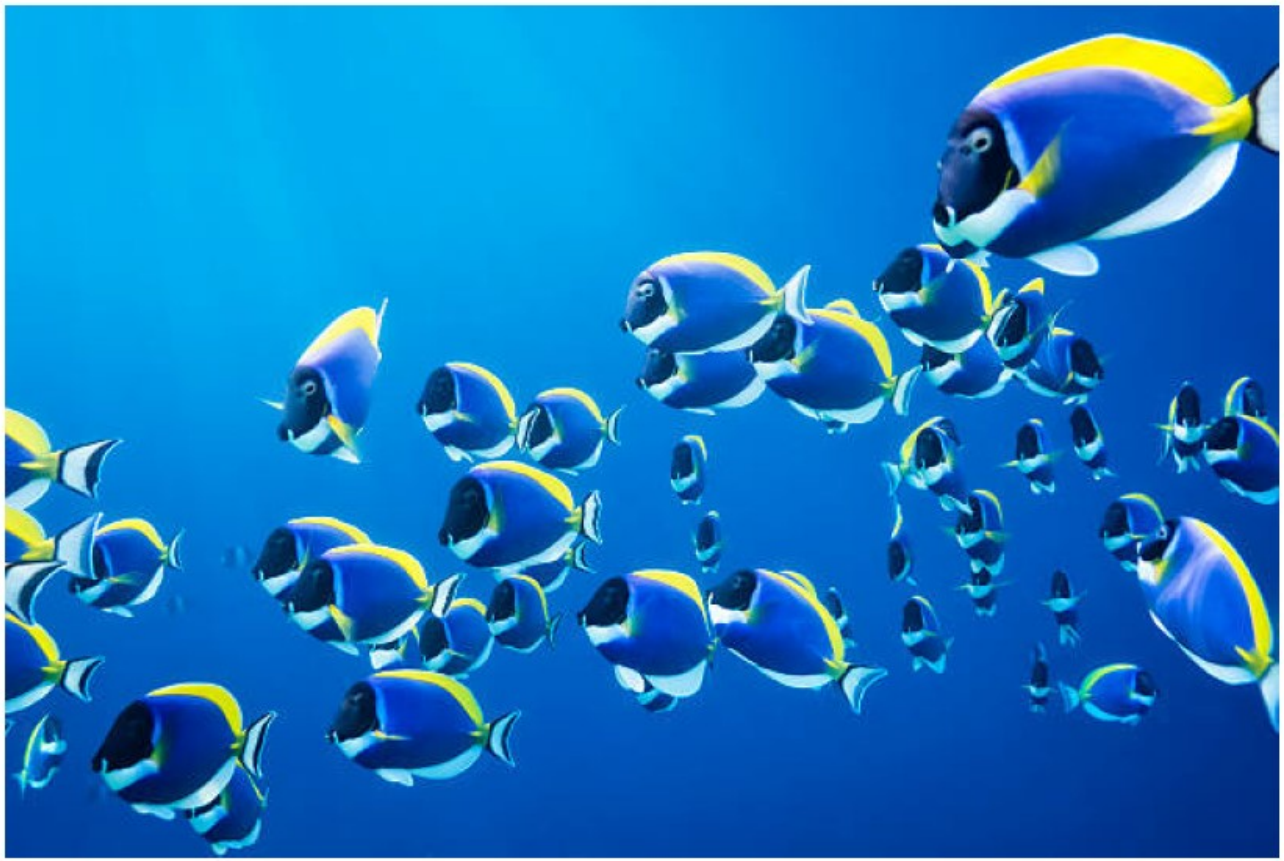
Entropy is an example of a texture feature

- One example is **entropy**: Measures the amount of information or randomness in the image region
- It is defined as:

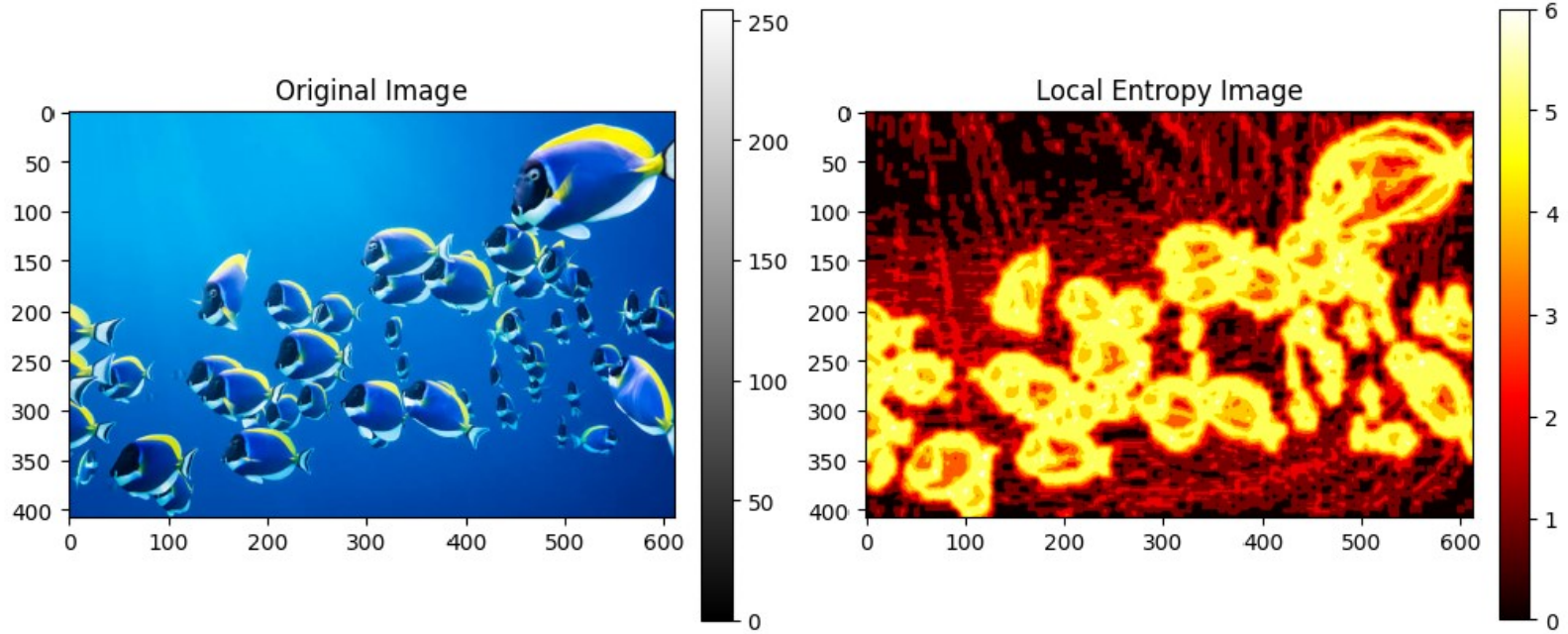
$$E = -\sum_{i,j} P(i,j) \log(P(i,j))$$

- $P(i,j)$ is the probability of co-occurrence of grayscale values i and j
- Higher entropy indicates a more complex texture with more details
- Lower entropy indicates a more uniform texture with fewer details

What heatmap of local entropy is expected for this image?



Local entropy is higher for the fish than the water



$\rightarrow$ Typical use cases: material classification, defect detection

Haar-like features can help to identify faces

- Haar-like features represent differences in intensity between rectangular regions of an image
- Calculated by summing the pixel intensities in specific regions and subtracting one region's sum from another

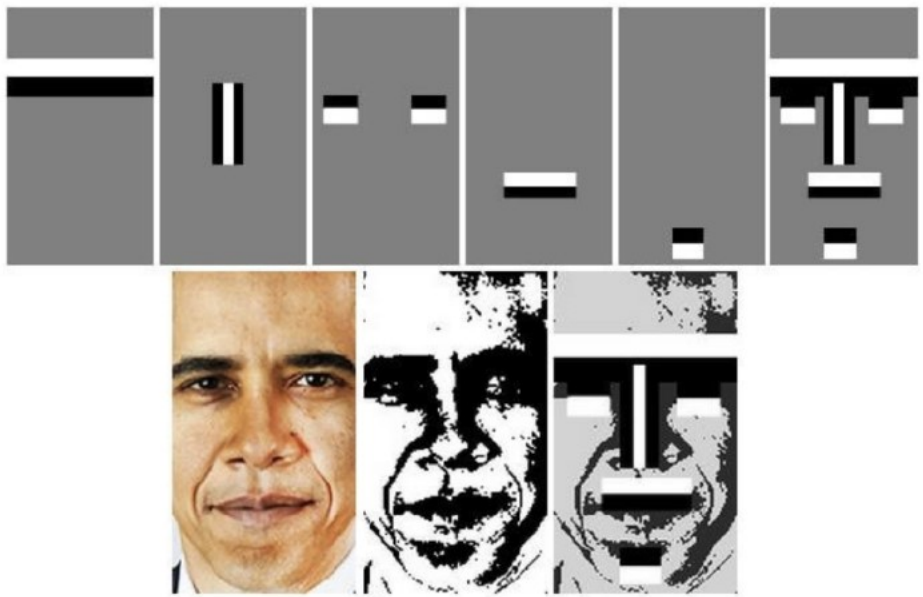


Figure: Kadir et al. (2014): Haar-like features for face detection.[5]

[5] Kadir, K., Kamaruddin, M. K., Nasir, H., Safie, S. I., and Bakti, Z. A. K. (2014). A Comparative Study between LBP and Haar-like Features for Face Detection using OpenCV. 4th International Conference on Engineering Technology and Technopreneuship (ICE2T) (pp. 335-339). IEEE

An image-based analysis pipeline looks like other classical ML-based pipelines

1. **Data collection and pre-processing**:
   - Gather a labeled dataset and pre-process the images (e.g., resize, convert to grayscale, normalize).
2. **Feature extraction and selection**:
   - Extract relevant features (e.g., texture, color, edges, key points) and choose the relevant ones (or reduce dimensionality).
3. **Model training**:
   - Train a model with the selected features (e.g., logistic regression, random forest).
4. **Model evaluation**:
   - Use metrics and cross-validation to assess model performance.
5. **Model deployment and prediction**:
   - Deploy the trained model for making predictions on (new) images.

Outline

Classical ML vs. Deep Learning

1. **Classical ML**:
   - **Manual Feature Engineering**: Requires domain expertise to design and extract features like edges, textures, shapes, or color histograms
   - **Separate Process**: Feature extraction is a distinct step performed before training the model
   - **Dependent on Domain Knowledge**: The effectiveness of the model depends heavily on the quality and relevance of manually selected features
2. **Deep Learning**:
   - **Automatic Feature Learning**: Pixels are basic features
   - **Integrated Process**: Features are learned directly from the data during training
   - **Hierarchical Features**: Learns low-level features (edges, corners) in early layers and high-level abstract features (object shapes, patterns) in deeper layers

Classic techniques: Graphical Illustration

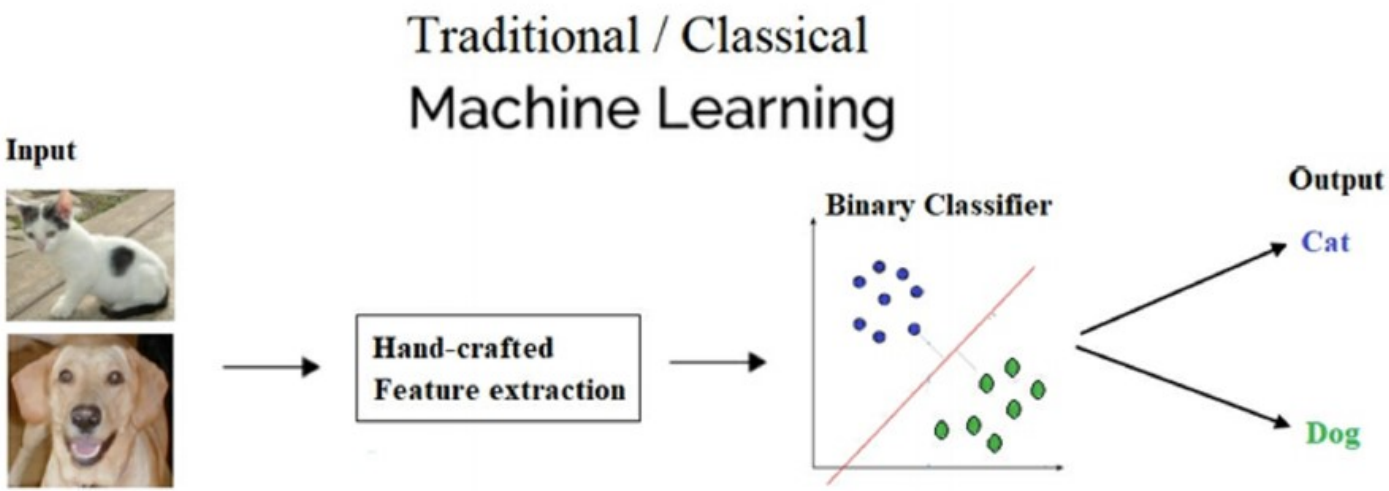


Figure: Dey (2018)[6]

[6] Dey, S. (2018). Hands-On Image Processing with Python: Expert Techniques for Advanced Image analysis and Effective Interpretation of Image Data. Packt Publishing Ltd.

Deep learning: Graphical Illustration

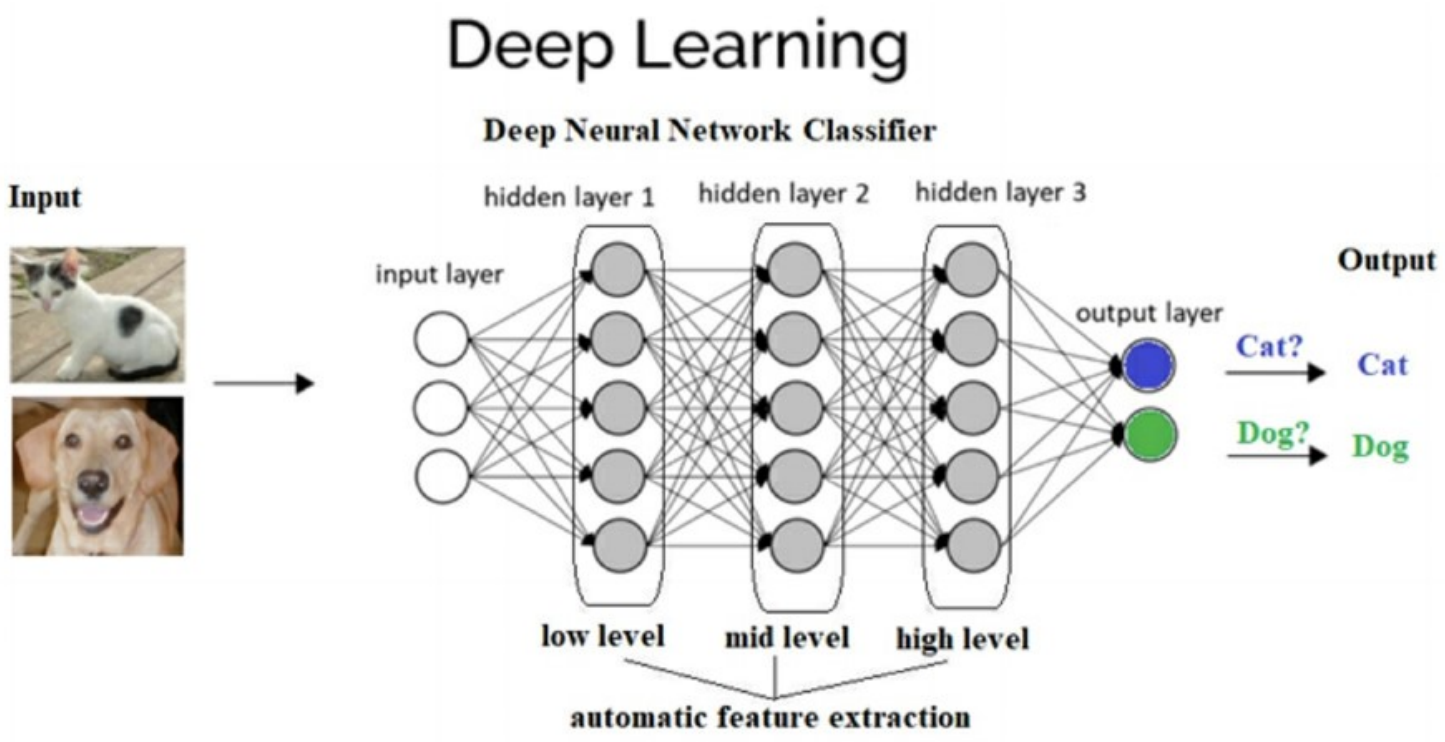


Figure: Dey (2018)[7]

[7] Dey, S. (2018). Hands-On Image Processing with Python: Expert Techniques for Advanced Image analysis and Effective Interpretation of Image Data. Packt Publishing Ltd.

CNNs are the current computer vision workhorse

- CNN is a specialized type of neural network designed to process and analyze structured data such as images
- Typically: input layer $\rightarrow$ hidden layers $\rightarrow$ output layer.
- What do different convolutional layers do?
  - Layers close to the input layer learn low-level features (e.g., lines)
  - Middle layers learn complex abstract features (combining lower-level features)
  - Layers closer to the output interpret the extracted features in the light of the classification task

CNN are traditionally employed in supervised settings

- CNNs are a specialized type of deep learning algorithm working with raw pixel values
- CNNs use many prelabeled training images to "learn" which pixel combinations are associated with the desired labels
- Once the performance is satisfactory, the algorithm can then be used to label large numbers of additional images quickly and at low cost
- Existing trained CNNs can be borrowed and fine-tuned (transfer learning) for (rather) accurate results using a smaller training set of images

Convolution is fundamental for the abstract representations in CNN

- Instead of simple matrix multiplication like in ANN, CNNs use convolutions
- A filter (kernel) slides over input data to produce a feature map
- Detects local patterns like edges and textures
- Non-linear activation functions (e.g., RELU) introduce non-linearity

Convolution is the basis of hierarchical feature learning

- Each filter (like K from earlier) produces a unique feature map:

$$\text{Image} \xrightarrow{FilterK} \text{Feature Map}$$

- A CNN layer typically contains multiple filters
- Accordingly, an input can produce multiple feature maps in one convolutional layer
- Over training, the network learns the optimal filter values to minimize the loss
- Pooling layers after convolution can down-sample the feature maps

Multi-channel image example

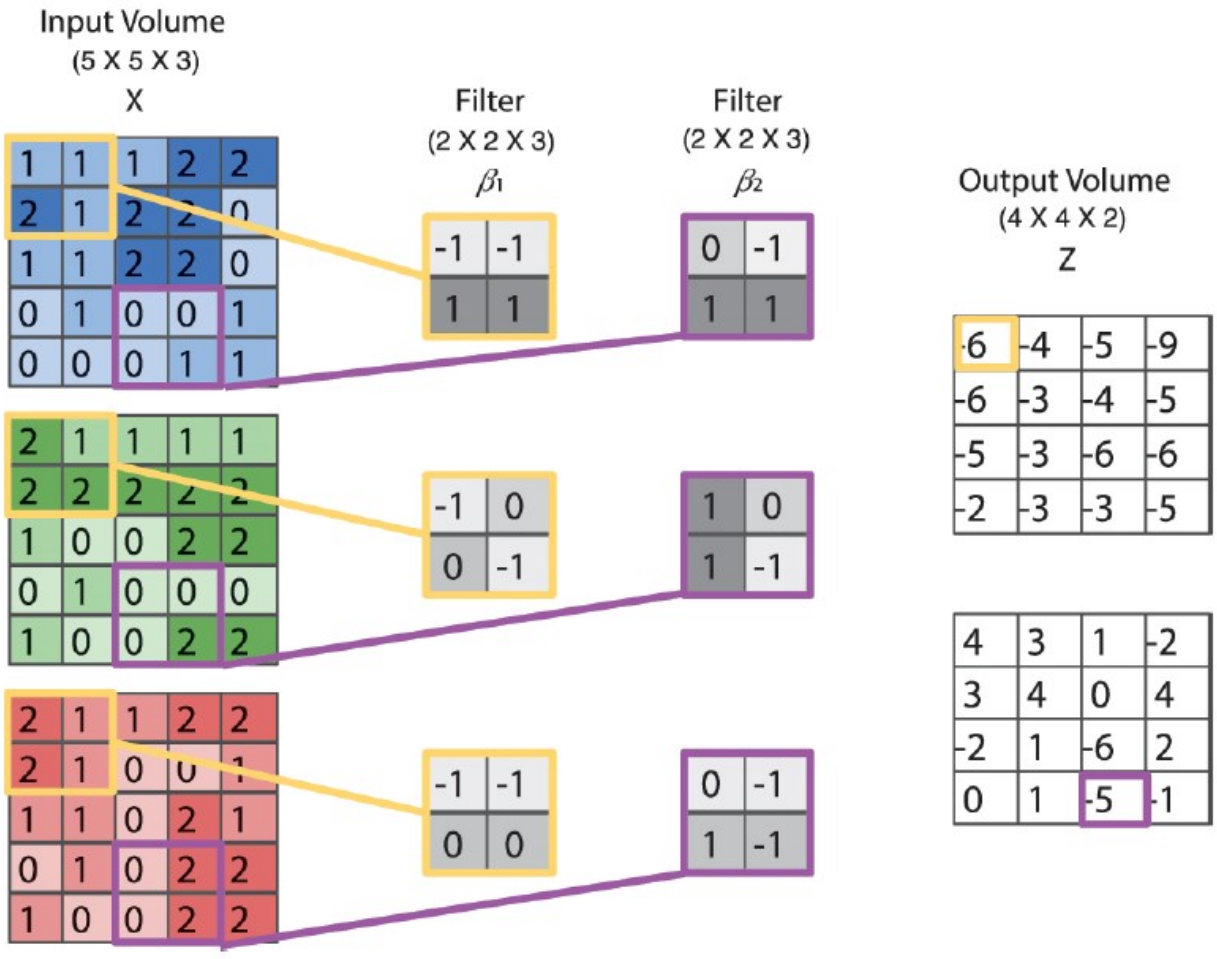


Figure: Convolutional Layer in a CNN for Image Processing (Webb Williams et al., 2020).

Steps of CNN Training (1/2)

- **Initialize Parameters:** Set weights and biases for all layers, typically randomly.
- **Input Forward Propagation:**
  - Pass input through convolutional, pooling, and fully connected layers.
  - Apply activation functions (e.g., ReLU, Softmax).
- **Loss Calculation:** Compare predictions with true labels using a loss function (e.g., cross-entropy).

Steps of CNN Training (2/2)

- **Backward Propagation:**
  - Compute gradients of the loss with respect to weights and biases using backpropagation.
  - Update weights through all layers via chain rule (from output to input).
- **Weight Update:** Adjust weights and biases using optimization algorithms (e.g., stochastic gradient descent).
- **Repeat:**
  - Iterate over the dataset for multiple epochs.
  - Evaluate on validation data to monitor performance and prevent overfitting.
- **Stop:** End training based on stopping criteria (e.g., convergence, maximum epochs).

CNN come with many hyperparameters

- **Epochs/iterations:** Number of times the model trains over the full set of images (e.g., 100 epochs)
- **Learning rate:** How much weights/coefficients change in optimization (e.g., 0.01)
- **Dropout rate:** To avoid overfitting; share of weights set to 0 (e.g., 0.5)
- **Batch size:** Number of samples processed before updating (e.g., 32 samples)
- **Train-validation ratio:** Split of images into sets (e.g., 80-20 split)
- **Loss function:** Evaluates model accuracy (e.g., Mean Squared Error)
- **Activation functions:** Nonlinear transformations (e.g., ReLU)
- More hyperparameters can be set (gamma, step size etc.)

Nonlinear transformations in deep learning

- Nonlinear transformations (activation functions) are applied to hidden layers
- Common activation functions: Sigmoid, Tanh, RELU

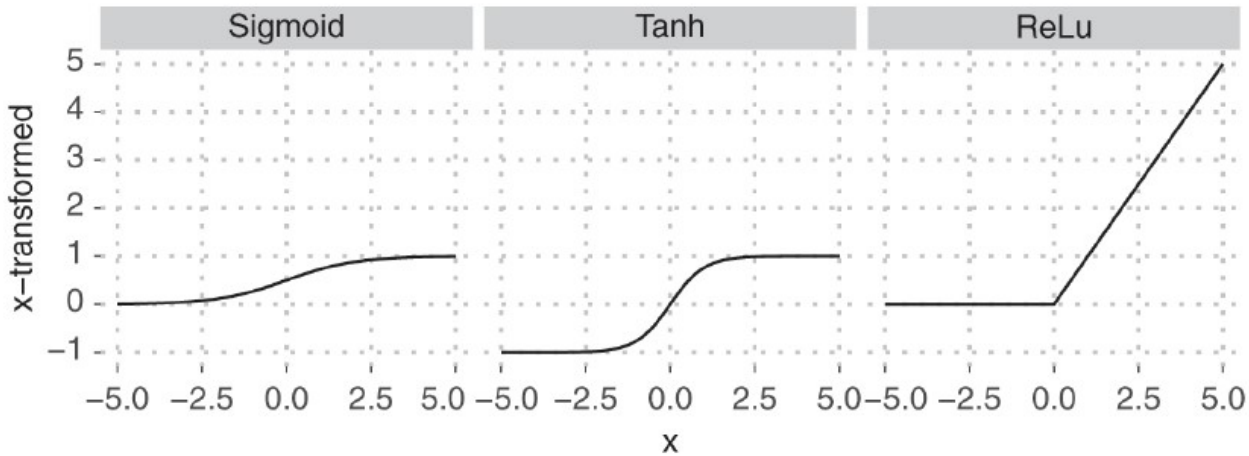


Figure: Three common nonlinear transformations (activation functions) in deep learning. Source: Webb Williams et al. (2020).

Example: CNN for predicting outcomes (1/4)

- CNN examples predicting outcomes using pixels from images are high dimensional
- Show low-dimensional toy example predicting outcomes (vote shares) using county characteristics (Webb Williams et al., 2020); can easily be adapted to high dimensional pixel case

County	White (%)	College (%)	Median Income (per capita, in hundreds USD)	Vote Share
Lake	87.70	16.20	215.37	0.46
Shasta	88.50	18.80	236.70	0.48
Mendocino	86.30	22.00	233.06	0.36
Sonoma	87.40	32.20	328.35	0.51
Sutter	74.00	18.70	236.02	0.55
Amador	90.70	19.30	273.47	0.52
Napa	84.80	31.30	347.95	0.60

Figure: Vote share for Hillary Clinton in the 2016 Democratic primaries in seven California counties

Example: CNN for predicting outcomes (2/4)

$$\underset{\text{(Input Layer: } 7\times 4)}{X}\begin{bmatrix}1 & 87.7 & 16.2 & 215.37\\ 1 & 88.5 & 18.8 & 236.70\\ 1 & 86.3 & 22.0 & 233.06\\ 1 & 87.4 & 32.2 & 328.35\\ 1 & 74.0 & 18.7 & 236.02\\ 1 & 90.7 & 19.3 & 273.47\\ 1 & 84.8 & 31.3 & 347.95\end{bmatrix}\quad \underset{\text{(Fully Connected Weights: } 4\times 3)}{\beta_1}\begin{bmatrix}-0.0007 & 0.6163 & -0.3102\\ -0.4425 & -0.1089 & 0.3600\\ -0.2725 & -0.5114 & 0.1802\\ -0.4045 & 0.1266 & 0.1269\end{bmatrix} = \underset{\text{(Hidden Layer 1 nontransformed: } 7\times 3)}{Z_0}\begin{bmatrix}-130.3434 & 10.0404 & 61.5057\\ -140.0340 & 11.3235 & 64.9685\\ -138.4599 & 9.4660 & 64.2914\\ -180.2716 & 16.1915 & 78.6155\\ -133.3151 & 12.8679 & 59.6443\\ -156.0177 & 15.4824 & 70.5158\\ -186.8042 & 19.4159 & 80.0041\end{bmatrix} \quad (1)$$

$$max(0, Z_0) = \underset{\text{(Hidden Layer 1 transformed)}}{Z_1}\begin{bmatrix}0 & 10.0404 & 61.5057\\ 0 & 11.3235 & 64.9685\\ 0 & 9.4660 & 64.2914\\ 0 & 16.1915 & 78.6155\\ 0 & 12.8679 & 59.6443\\ 0 & 15.4824 & 70.5158\\ 0 & 19.4159 & 80.0041\end{bmatrix} \quad (2)$$

Figure: Convolutional neural network predicting Clinton's vote share $\hat{Y}$: (1) using input layer X to create a new hidden layer Z_0, (2) applying a ReLu transformation to Z_0, (Webb Williams et al., 2020).

Example: CNN for predicting outcomes (3/4)

$$\underset{\text{(Hidden Layer 1 transformed)}}{Z_1} = \begin{bmatrix} 0 & 10.0404 & 61.5057 \\ 0 & 11.3235 & 64.9685 \\ 0 & 9.4660 & 64.2914 \\ 0 & 16.1915 & 78.6155 \\ 0 & 12.8679 & 59.6443 \\ 0 & 15.4824 & 70.5158 \\ 0 & 19.4159 & 80.0041 \end{bmatrix} * \underset{\text{(Convolutional Filter: } 1 \times 2)}{\beta_2} \begin{bmatrix} -0.5090 & 0.186 \end{bmatrix} = \underset{\text{(Hidden Layer 2)}}{K} \begin{bmatrix} 1.8729 & 6.3629 \\ 2.1122 & 6.3558 \\ 1.7657 & 7.1749 \\ 3.0203 & 6.4238 \\ 2.4003 & 4.5766 \\ 2.8880 & 5.2738 \\ 3.6218 & 5.0418 \end{bmatrix} \quad (3)$$

Figure: ...(3) using a 1 x 2 convolutional filter to create a new hidden layer K...

Example: CNN for predicting outcomes (4/4)

$$\underset{\text{(Hidden Layer 2: } 7\times 3)}{K} \times \underset{\text{(Fully Connected Weights: } 3\times 1)}{\beta_3} = \underset{\text{(Output Layer nontransformed: } 7\times 1)}{\hat{Y}_0} \tag{4}$$

$$\begin{bmatrix} 1 & 1.8729 & 6.3629 \\ 1 & 2.1122 & 6.3558 \\ 1 & 1.7657 & 7.1749 \\ 1 & 3.0203 & 6.4238 \\ 1 & 2.4003 & 4.5766 \\ 1 & 2.8880 & 5.2738 \\ 1 & 3.6218 & 5.0418 \end{bmatrix} \times \begin{bmatrix} 0.4660 \\ 0.2257 \\ -0.1782 \end{bmatrix} = \begin{bmatrix} -0.2450 \\ -0.1897 \\ -0.4139 \\ 0.0031 \\ 0.1923 \\ 0.1782 \\ 0.3852 \end{bmatrix}$$

$$\frac{1}{(1+\epsilon^{-\hat{Y}_0})} = \underset{\text{(Output Layer transformed)}}{\hat{Y}_1} \tag{5}$$

$$\begin{bmatrix} 0.4391 \\ 0.4527 \\ 0.3980 \\ 0.5008 \\ 0.5479 \\ 0.5444 \\ 0.5951 \end{bmatrix}$$

Figure: ...(4) using K to generate a set of predictions ($\hat{Y}_0$), and (5) applying a sigmoid transformation to improve model fit.

Key Takeaways

- CNNs are highly effective for image-based tasks
- Understanding hyperparameters is key for optimizing CNNs
- Data preparation and labeling remain challenges
- Ethical and computational considerations remain critical

Outline

Auto-Taggers for Image Classification

- Use pre-trained CNN models to extract features and predict image tags.
- Assign tags by:
  - Selecting the most likely category for single-label tasks.
  - Selecting multiple categories with high confidence scores for multi-label tasks.
- Fine-tune predictions by applying simple rules, such as ignoring unlikely combinations of tags.

Pros and Cons of Autotaggers (vs. Fine-Tuning CNNs)

Pros:

- Faster setup and deployment, as it skips the need for full training
- Works well out-of-the-box for common tagging tasks

Cons:

- Cannot adapt easily to specialized or highly unique image datasets
- Might struggle to capture subtle patterns without retraining on specific examples

Vision Transformers (Vits) for Image Classification: Overview

- Deep learning architecture for image classification and other vision tasks, inspired by the Transformer models commonly used in natural language processing
- Unlike CNNs, which rely on convolutional operations to extract features, ViTs use self-attention mechanisms to capture relationships between image patches
- Self-attention calculates how much every element (e.g., a patch in an image) is influenced by other elements
- Assigns weights (attention scores) to these relationships, emphasizing important elements and de-emphasizing less relevant ones

Vision Transformers (Vits) for Image Classification: Steps

- Divide the input image into small patches (e.g., 16×16) and flatten each patch into a vector.
- Embed each patch vector with position information to preserve spatial relationships.
- Use Transformer architecture to process patches:
  - Self-attention layers capture global dependencies between patches.
  - Feed-forward layers refine features.
- Output a classification token summarizing the image for prediction.

Vision Transformer: Graphical Illustration

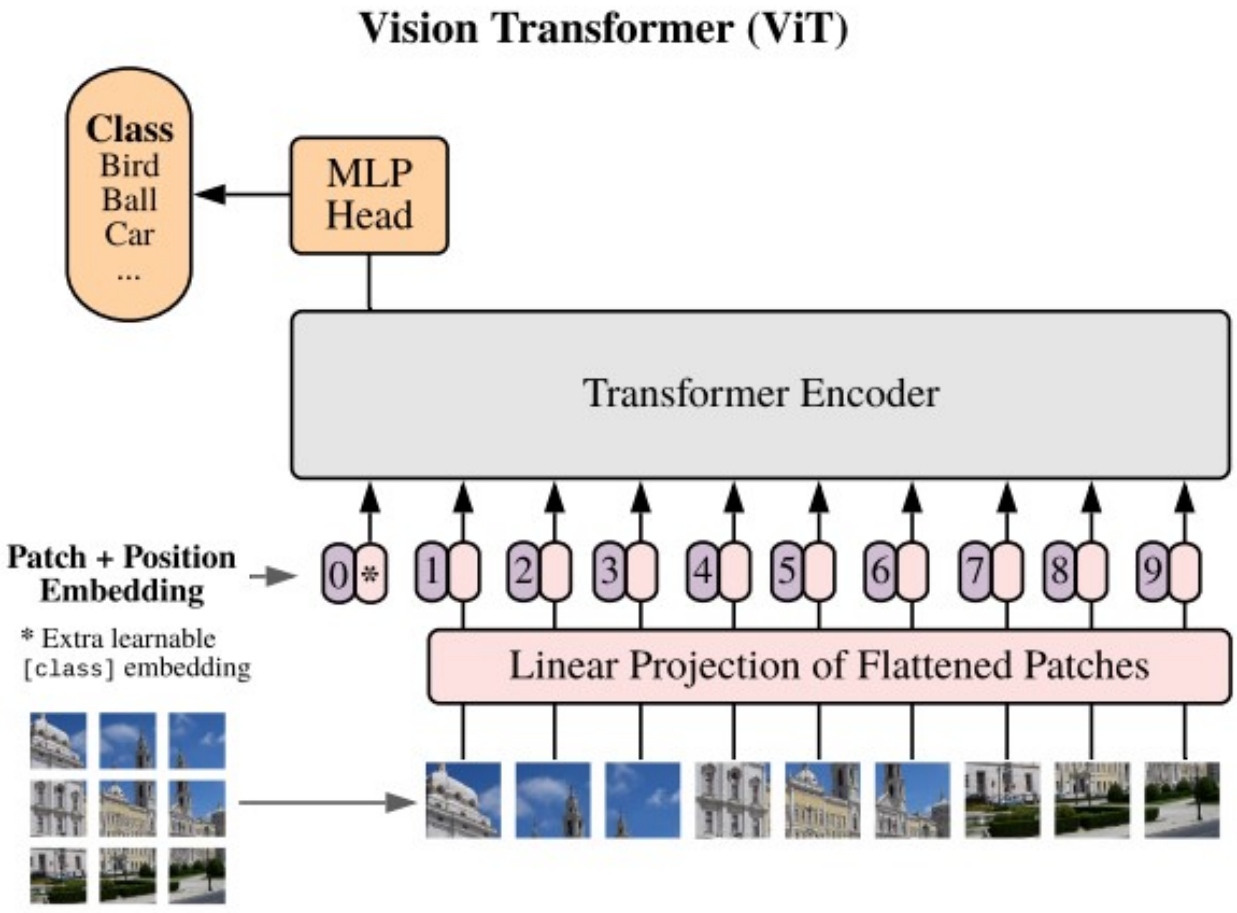


Figure: Vision Transfomer Model from "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" (Dosovitskiy et al, 2021)

Pros and Cons of Vision Transformers (vs. CNNs)

Pros:

- Better at capturing global relationships across an image
- Scales well with large datasets and complex tasks
- Leverages pre-trained models from natural language processing (e.g., BERT-like architectures)

Cons:

- Requires large datasets for effective training
- Computationally intensive, especially for high-resolution images
- May struggle with small datasets compared to CNNs without significant tuning