2.Interpretability of Neural Networks

CNN Feature learning

- CNNs learn hierarchical, abstract features directly from raw pixels
- Eliminates traditional feature engineering (edge detectors, color histograms, etc.)
- Features become increasingly complex through network depth

Traditional ML vs. Deep Learning:

Traditional (e.g., SVM for images):

- Manual feature engineering required
- Extract color features, frequency domain, edge detectors
- Feed engineered features to classifier

Deep Learning (CNNs):

- Raw pixel input → Network learns features automatically
- Convolutional layers extract features hierarchically
- Fully connected layers map features to predictions
- Key Advantage: Network discovers optimal features for the task

What Features Are Learned at Each Layer?

Layer-by-Layer Progression:
First Convolutional Layer(s):
Edges (horizontal, vertical, diagonal)
Simple textures
Basic colors and color combinations
Middle Convolutional Layers:
More complex textures and patterns
Combinations of edges
Simple geometric shapes
Later Convolutional Layers:
Objects or parts of objects (dog snouts, wheels, eyes)
Complex patterns
Scene elements
Fully Connected Layers:
Connect high-level features to class predictions
Learn which combinations indicate which classes

Feature Visualization

Feature Visualization finds the input that maximizes the activation of a neural network unit.

What is a "Unit"?
Can refer to different granularities:
1. Individual Neurons
    - Most atomic, most information
    - Problem: Millions of neurons to visualize!
2. Channels (Feature Maps)
    - Good compromise between detail and feasibility
    - Each channel detects specific patterns
3. Entire Layers
    - Used in Google DeepDream
    - Higher-level abstraction
4. Final Class Neuron
    - Pre-softmax neuron recommended
    - Shows what maximally activates a class

Goal: Generate an image that makes a unit "fire" maximally

Mathematical formulation of feature visualization

For a Single Neuron:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} h_{n,u,v,z}(\mathbf{x})$$

Where:
- x* = optimal input image we're searching for
- h = activation function
- n = layer index
- u, v = spatial position of neuron
- z = channel index

For an Entire Channel (Mean Activation):

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \sum_{u,v} h_{n,u,v,z}(\mathbf{x})$$

Interpretation:
- Sum over all spatial positions (u, v)
- All neurons in channel z are equally weighted
- Find image that maximizes average channel activation

Positive and Negative Activations

Units respond to different features when maximizing vs. minimizing activation

Example – For A Wheel Detector Channel:

Positive Activation (Maximize):

Shows wheels, circular objects, What the channel is "looking for"

Negative Activation (Minimize):

Shows features with eyes, organic shapes, What the channel actively "avoids"

Generation of Feature Visualizations

Approach 1: Search Through Training Images
Method: Select existing images that maximally activate the target unit
Major Problem: Features in natural images are highly correlated

Example: If images contain both dogs and tennis balls, we cannot determine which feature activates the unit.
Impossible to isolate what the network truly detects

Approach 2: Generate New Images

- Start with completely random noise as the initial image
- Apply gradient ascent optimization iteratively
- Gradually modify pixels to progressively increase activation

Optimization Process Steps:
Initialize: Begin with a random noise image
Forward Pass: Compute the activation of the target unit
Backward Pass: Calculate gradient $\partial(\text{activation})/\partial(\text{pixels})$
Update: Adjust pixels in the direction of the gradient
Iterate: Repeat steps 2-4 until convergence is achieved

Making feature visualizations interpretable

- Basic optimization without constraints produces noisy, uninterpretable images
- High-frequency patterns and artifacts dominate the resulting visualizations
- Images often contain adversarial-like patterns that humans cannot understand

Regularization Techniques:

1. Image Transformations (Applied During Optimization):

- **Jittering**: Apply slight random translations to the image
- **Rotation**: Introduce small random rotations
- **Scaling**: Randomly zoom in or out slightly

Purpose: Makes visualizations more robust to minor perturbations

2.Frequency Penalization:

- Reduce the variance between neighboring pixels
- Add penalties for high-frequency noise components
- Encourages generation of smoother, more natural-looking images

3. Learned Priors from Generative Models:

- GANs (Generative Adversarial Networks): Constrain outputs to realistic image distributions
- Denoising Autoencoders: Remove noise while preserving meaningful features
- Effect: Restricts search space to images that resemble natural photographs

Feature visualization vs Adversarial examples

- Both techniques employ gradient-based optimization to maximize activation of specific neurons
- Both utilize gradient-based optimization with respect to input pixels
- Adversarial examples reveal network vulnerabilities and weaknesses
- Feature visualization reveals learned internal representations

Aspect	**Feature Visualization**	**Adversarial Examples**
Target	Any unit (neuron, channel, layer)	Specifically targets incorrect class neuron
Starting Point	Random noise image	Correctly classified image
Primary Goal	Understand what network learned	Intentionally fool or deceive the network
Constraints	Generate interpretable image	Remain visually similar to original
Purpose	Interpretability and understanding	Security testing and robustness evaluation

Network Dissection

- Feature visualizations rely on subjective human interpretation
- No existing quantitative measure of network interpretability
- Need to test hypothesis: Do networks truly learn disentangled concepts?

Network Dissection Method (Bau et al., 2017):

- Systematically links highly activated CNN channels with human-labeled concepts
- Provides objective, numerical measure of concept alignment

Core Research Questions:

Does channel 394 specifically detect skyscrapers?
Does channel 121 specifically detect dog snouts?
How well does this alignment work? (quantifiable with IoU score)

Key Innovation:

- Provides numeric interpretability score (Intersection over Union) for each unit
- Enables objective comparison of interpretability across different models

Network Dissection Algorithm

Three Steps to Quantify Interpretability

Step 1: Obtain Labeled Data

- Collect images with pixel-wise human concept labels
- Labels must span multiple abstraction levels (from colors to objects to scenes)
- Broden dataset was specifically created for this purpose

Step 2: Measure CNN Activations

- Forward propagate labeled images through the target network
- Extract activation maps for all channels
- Create binary masks identifying highly activated regions

Step 3: Quantify Alignment with Concepts

- Compare activation masks with ground truth concept labels
- Calculate Intersection over Union (IoU) metric for alignment
- Determine whether each channel qualifies as a concept detector

Final Output:

- For each channel: IoU score indicating alignment with best-matching human concept
- Enables ranking of channels by interpretability

Network Dissection Stepwise in detail

Step 1 - The Broden Dataset

The Challenge:

- Network Dissection requires images with pixel-level concept annotations
- Annotations must span multiple levels of abstraction
- Such datasets are extremely labor-intensive to create

Broden Dataset Composition:

- Contains 60,000 images in total
- Includes over 1,000 distinct visual concepts across hierarchical levels:

Annotation Methodology:

- Primarily uses pixel-wise segmentation masks
- Some datasets provide whole-image labels
- Requires extensive manual annotation effort

Step 2 - Activation Mask Creation

Extracting Highly Activated Regions

Detailed Algorithm for Each Convolutional Channel k:

Phase 1: Collect Activation Statistics Across Dataset

- Forward propagate all Broden images through the network
- For each image x, extract activation map: $A_k(x)$
- Aggregate activation distribution α_k across all images

Phase 2: Determine Activation Threshold

- Calculate the 0.995-quantile of all activations: T_k
- Interpretation: Only the top 0.5% of activations exceed this threshold
- This threshold identifies "highly activated" pixels for the channel

Phase 3: Create Binary Masks for Each Image

- Scale activation map $A_k(x)$ to match original image resolution → $S_k(x)$
- Apply binarization: $M_k(x) = [S_k(x) \geq T_k]$
- Result: Binary mask where 1 indicates highly activated pixels, 0 indicates low activation

Output of Step 2:

- One binary activation mask $M_k(x)$ for each channel k and each image x
- These masks will be compared against concept labels in Step 3

Step 3 - Measuring Alignment with IoU

Title: Intersection over Union (IoU) Score

The IoU Formula:

$$IoU_{k,c} = \frac{| M_k(\mathbf{x}) \cap L_c(\mathbf{x}) |}{| M_k(\mathbf{x}) \cup L_c(\mathbf{x}) |}$$

Component Definitions:

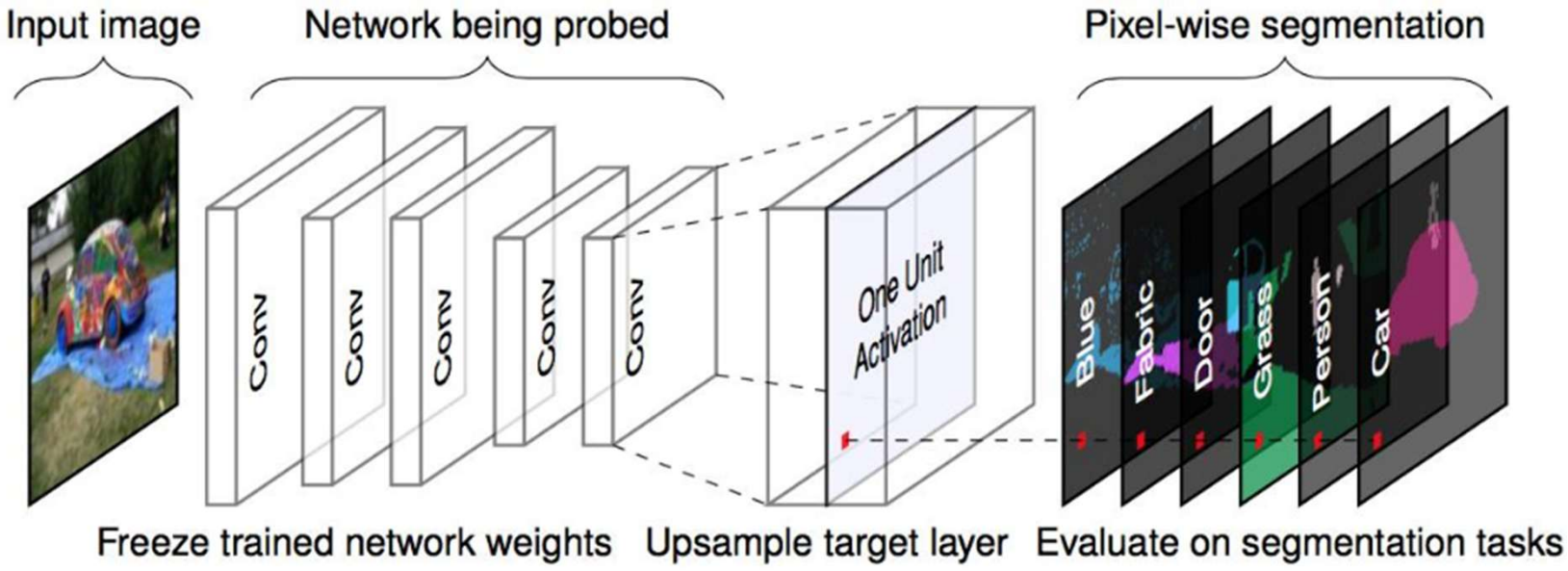
- M_k(x) = Binary activation mask for channel k on image x
- L_c(x) = Binary ground truth segmentation mask for concept c on image x
- |·| = Cardinality operator (counts number of pixels in the set)
- ∩ = Intersection operator (pixels active in both masks)
- ∪ = Union operator (pixels active in either mask)

IoU Score Interpretation:

- IoU = 1.0 indicates perfect alignment between activation and concept
- IoU = 0.5 indicates 50% overlap between activation and concept
- IoU = 0.0 indicates no overlap (no alignment)

Decision Rule for Concept Detection:

- Channel k is classified as a detector for concept c if: IoU_{k,c} > 0.04
- This threshold of 0.04 was determined empirically by Bau et al.
- Each channel is assigned to its highest-IoU concept as its primary detection target



Network Dissection Algorithm

Strengths of Feature Visualization

Strength 1: Unique Insight into Network Internals
•Only method that directly visualizes what internal units have learned
•Provides complementary information to other interpretability methods
•Offers visual, intuitive understanding of network representations

Strength 2: Reveals Hierarchical Learning Process
•Empirically confirms the progression: edges → textures → parts → objects
•Demonstrates how networks build abstractions layer by layer
•Helps explain why deep learning achieves strong performance

Strength 3: Accessible Non-Technical Communication
•Easy to explain to stakeholders without machine learning expertise
•Does not require mathematical or ML background to understand
•Excellent for presentations, reports, and public communication

Strength 4: Synergy with Attribution Methods
•Attribution methods (e.g., SHAP, LIME): "These specific pixels were important for this prediction"
•Feature visualization: "Here's what the network detected in those pixels"
•Combined approach provides complete explanation of model behavior

Why Use Network Dissection?

Strength 1: Quantifiable Interpretability Metric
•Provides objective numeric IoU scores rather than subjective assessments
•Enables rigorous comparison of interpretability across different models
•Supports scientific measurement instead of relying on intuition alone

Strength 2: Automated Concept Linking
•Eliminates need for manual inspection of thousands of individual channels
•Algorithm automatically identifies the best-matching concept for each unit
•Scales efficiently to large networks with many layers

Strength 3: Detects Concepts Beyond Training Classes
Network trained on 1,000 ImageNet classes can detect 1,000+ Broden concepts
Reveals that networks learn richer representations than strictly required for task
Shows transfer learning potential and generalization capability

Strength 4: Comprehensive Systematic Analysis
Analyzes entire network comprehensively rather than cherry-picking examples
Identifies which layers learn which types of concepts
Provides objective measurement of feature disentanglement across network

Use Feature Visualization When:

- Debugging neural networks to understand why specific predictions failed
- Conducting research on representation learning mechanisms
- Comparing learned features across different network architectures
- Understanding what concepts contributed to specific individual predictions
- Communicating neural network behavior to non-technical audiences

Use Network Dissection When:

- Need quantitative, objective measure of network interpretability
- Comparing interpretability of multiple models rigorously
- Studying effects of hyperparameters, architectures, or training procedures
- Conducting research on feature disentanglement
- Performing systematic bias detection across the network

Saliency Maps

- Pixel attribution methods highlight which pixels were relevant for a neural network's image classification
- Also known as: sensitivity maps, saliency maps, gradient-based attribution, feature relevance

- Special case of feature attribution specifically designed for images
- Explains individual predictions by showing how much each pixel contributed to the prediction
- Produces visual explanations the same size as the input image

Input: Image x with p pixels
Network output: S(x) = prediction vector of length C classes
Output: Relevance score R^c for each pixel indicating importance for class c

2 Categories of Saliency Maps
Category 1: Occlusion/Perturbation-Based Methods
•Manipulate or hide parts of the image to see effect on prediction
•Examples: SHAP, LIME
•Model-agnostic (work with any black-box model)
•Computationally expensive

Category 2: Gradient-Based Methods

•Compute gradient of prediction with respect to input pixels
•Leverage neural network's internal gradient information
•Multiple methods differ primarily in how gradients are computed
•Faster than occlusion-based methods
•Model-specific (require gradient access)

Two Philosophies of Gradient Methods

Gradient-Only Methods:

- Answer: "If I change this pixel, how does prediction change?"
- Examples: Vanilla Gradient, Grad-CAM
- Interpretation: Positive gradient = increasing pixel value increases predicted probability
- Larger absolute gradient = stronger effect of pixel change
- No baseline reference needed

Path-Attribution Methods:

- Answer: "How does this image differ from a baseline reference?"
- Examples: Integrated Gradients, Deep Taylor, SHAP, LIME
- Compare current image to reference (e.g., gray image, blurred image, dataset distribution)
- Attribute difference between actual and baseline predictions across pixels
- Some are "complete": sum of relevances = prediction difference from baseline

Key Difference:

Gradient-only: Local sensitivity analysis
Path-attribution: Comparison to reference state

Vanilla Gradient

Core Idea
Calculate gradient of class score with respect to input pixels
Produces map showing which pixels most affect the prediction
First major pixel attribution approach

$$E_{grad}(\mathbf{x}_0) = \frac{\delta S_c}{\delta \mathbf{x}}|_{\mathbf{x}=\mathbf{x}_0}$$

Where:
•Sc = score for class c
•x = input image pixels
•Gradient shows rate of change of class score with respect to each pixel

Algorithm Steps:
1.Forward pass image through network
2.Compute gradient of target class score w.r.t. input pixels (set other classes to zero)
3.Visualize gradients (absolute values or separate positive/negative contributions)

The Saturation Issue:

- When ReLU is used and activation goes below zero:
- Activation capped at zero (does not change anymore)
- Activation is saturated
- Gradient at saturation point becomes zero

Problem: Vanilla Gradient incorrectly indicates neuron is not important

Example Scenario:

Consider layer with:
Two input neurons with weights: -1 and -1
Bias: +1
Followed by ReLU
Behavior:
If neuron1 + neuron2 < 1: activation = neuron1 + neuron2
If neuron1 + neuron2 > 1: activation saturated at 1 (negative weights cause saturation)
At saturation: gradient = 0

Consequence:

- Vanilla Gradient fails to capture importance of saturated neurons
- Misses important features when neurons operate in saturation regime

Grad-CAM (Gradient-weighted Class Activation Mapping)

Goal: Understand which parts of an image a convolutional layer "looks" at for classification
Key Innovation: Gradient backpropagated only to last convolutional layer (not to input pixels)
•Produces coarse localization map
•Highlights important regions rather than individual pixels

Why Last Convolutional Layer?
•Contains high-level semantic information
•Retains spatial information about object locations
•Has k feature maps: A^1, A^2, ..., A^k
•Feature maps encode information for all classes

Problem to Solve:
•Feature maps contain information for all classes
•We're interested in specific class c
•Need to weight each feature map by its importance to class c

Solution Approach:
•Weight each pixel of each feature map with gradient
•Average over feature maps
•Apply ReLU to keep only positive contributions (parts contributing to class c)

$$L^c_{\text{Grad-CAM}} \in \mathbb{R}^{U \times V} = \underbrace{\text{ReLU}}_{\text{Pick positive values}} \left(\sum_k \alpha_k^c A^k \right)$$

Where:
•L = localization map of size U × V (width × height)
•α_k^c = importance weight for feature map k and class c
•A^k = k-th feature map
•ReLU = keeps only positive values (features contributing to class c)

$$\alpha_k^c = \overbrace{\frac{1}{Z} \sum_u \sum_v}^{\text{global average pooling}} \underbrace{\frac{\delta y^c}{\delta A_{uv}^k}}_{\text{gradients via backprop}}$$

•Global average pooling of gradients
•Averages gradient over all spatial locations (u, v)

Guided GradCAM

- Grad-CAM provides coarse localization (which regions matter)
- Other methods (e.g., Vanilla Gradient) provide fine pixel-level detail
- Both have complementary strengths

The Fusion Approach:

Combines Grad-CAM with pixel-level attribution method

Gets "best of both worlds": location + detail

Algorithm:

- Compute Grad-CAM explanation for image
- Compute pixel-level explanation (e.g., Vanilla Gradient, Guided Backpropagation)
- Upsample Grad-CAM output using bilinear interpolation to match image size
- Element-wise multiply both explanation maps

Effect:

- Grad-CAM acts as a "lens" focusing on specific regions
- Pixel-level detail shown only in Grad-CAM-highlighted regions
- Combines spatial localization with fine-grained attribution

Benefits:

- More interpretable than either method alone
- Shows both "where" (Grad-CAM) and "what" (pixel method)
- Reduces noise outside regions of interest