MIDDLE EAST TECHNICAL UNIVERSITY

SEMESTER I EXAMINATION 2024-2025

CENG 403 – Deep Learning - CNN Visualization & Classic Architectures - ANSWERED

January 2025 TIME ALLOWED: 3 HOURS

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **SIX (6)** questions and comprises **EIGHT (8)** printed pages.
2. Answer all questions. The marks for each question are indicated at the beginning of each question.
3. Answer each question beginning on a **FRESH** page of the answer book.
4. This **IS NOT an OPEN BOOK** exam.
5. Show clear reasoning for your answers, especially intuitive explanations.
6. For architectural diagrams, draw clear components and explain design choices.
7. Connect concepts to examples discussed in lectures where relevant.
8. Explain the practical implications of design decisions.

Question 1. Class Activation Maps and Global Average Pooling (25 marks)
Based on the professor's explanation: "Each channel actually specializes in capturing one part or maybe the whole part of an object... it functions as a confidence map over pixels."

(a) The professor explained that global average pooling forces each channel to specialize. Explain how class activation maps exploit this property to visualize what the network is "paying attention to." Include the mathematical process of weighting channels. (10 marks)

Answer: Class Activation Maps (CAM) exploit the channel specialization property of Global Average Pooling to create spatial visualizations showing which regions the network considers important for each class.

Channel Specialization with Global Average Pooling:

When using Global Average Pooling, each channel in the final convolutional layer learns to specialize as a "confidence map" for detecting specific semantic concepts. This happens because:

The final classification layer uses weights $w_{k,c}$ to combine channel activations:

$$y_c = \sum_{k=1}^{K} w_{k,c} \cdot \text{GAP}(f_k)$$

where $\text{GAP}(f_k) = \frac{1}{H \times W} \sum_{i,j} f_k(i,j)$ is the global average of channel k.

To maximize class c score, the network learns to make channel k have high activations wherever objects relevant to class c appear, and low activations elsewhere.

Class Activation Map Generation:

For a given class c, the CAM is computed as:

$$\text{CAM}_c(i,j) = \sum_{k=1}^{K} w_{k,c} \cdot f_k(i,j)$$

This creates a spatial heatmap where:

- High values indicate regions important for class c

- Each pixel's importance is a weighted combination of all channel activations
- The weights $w_{k,c}$ are learned during training for classification

Mathematical Process: 1. Forward pass through CNN to get final conv layer activations $f_k(i, j)$ 2. Use learned classification weights $w_{k,c}$ 3. Compute weighted sum across channels for each spatial location 4. Upscale to input image resolution if needed 5. Overlay on original image as heatmap

The key insight is that GAP forces channels to become semantic detectors, and CAM visualizes how these detectors vote for each class across the image.

(b) Compare class activation maps with Grad-CAM as described by the professor. Explain why Grad-CAM replaces learned weights with "summation of gradients with respect to the channel" and how this makes the visualization "input dependent." (10 marks)

Answer: Grad-CAM generalizes CAM by replacing fixed learned weights with input-specific gradient information, enabling visualization for any CNN architecture without requiring Global Average Pooling.

Key Differences Between CAM and Grad-CAM:

CAM Limitations:

- Requires Global Average Pooling architecture
- Uses fixed learned weights $w_{k,c}$ from classification layer
- Cannot be applied to networks with fully connected layers
- Same importance weights used regardless of input image

Grad-CAM Solution:

Instead of using learned weights $w_{k,c}$, Grad-CAM computes:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{i,j}^k}$$

where:

- y^c is the score for class c
- $A^k_{i,j}$ is the activation at spatial location (i, j) in channel k
- α^c_k represents the importance of channel k for class c for this specific input

The Grad-CAM heatmap is:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha^c_k A^k\right)$$

Why Gradients Make It Input-Dependent:

The gradients $\frac{\partial y^c}{\partial A^k_{i,j}}$ capture how much a small change in channel k's activation would affect the class c score for the current input. This means:

- Different inputs produce different gradient patterns
- The importance weights α^c_k adapt to what the network is actually using for this specific prediction
- More relevant channels get higher weights for inputs where they're discriminative
- Less relevant channels get lower weights when they don't contribute to the decision

Practical Advantages:

(a) **Architecture Flexibility:** Works with any CNN, including those with FC layers

(b) **Input Specificity:** Highlights features actually used for each prediction

(c) **Layer Flexibility:** Can visualize any convolutional layer, not just the last one

(d) **Better Localization:** Often provides more precise localization than CAM

The gradient information tells us "for this specific input, how important is each channel for the predicted class," making the visualization much more informative and reliable.

(c) The professor mentioned that "if you do this with a CNN that has fully connected layers you are not going to get such activation maps." Explain why global average pooling is essential for class activation maps and why fully connected layers prevent this visualization. (5 marks)

Answer: Fully connected layers destroy spatial information by flattening feature maps into vectors, making it impossible to trace back class-specific importance to spatial locations in the original image.

Why Global Average Pooling Enables CAM:

GAP preserves the connection between spatial locations and class predictions:

- Each channel $f_k(i, j)$ maintains spatial coordinates (i, j)
- The classification layer directly uses channel averages: $y_c = \sum_k w_{k,c} \cdot \text{GAP}(f_k)$
- We can "reverse" this process: $\text{CAM}_c(i, j) = \sum_k w_{k,c} \cdot f_k(i, j)$
- This shows exactly which spatial locations contribute to each class score

Why Fully Connected Layers Break CAM:

FC layers perform: $h = W \cdot \text{flatten}(f) + b$

This creates several problems:

(a) **Spatial Information Loss:** Flattening converts $f_k(i, j)$ into a 1D vector, losing all spatial structure

(b) **Mixed Spatial Responses:** Each FC neuron receives input from ALL spatial locations across ALL channels

(c) **No Direct Mapping:** Cannot trace how specific spatial locations influence the final prediction

(d) **Complex Interactions:** Multiple FC layers create complex non-linear transformations that obscure spatial contributions

Mathematical Explanation:

With GAP: $y_c = \sum_k w_{k,c} \cdot \frac{1}{HW} \sum_{i,j} f_k(i, j)$

This can be rewritten as: $y_c = \frac{1}{HW} \sum_{i,j} \sum_k w_{k,c} \cdot f_k(i, j) = \frac{1}{HW} \sum_{i,j} \text{CAM}_c(i, j)$

With FC layers: $y_c = g(\text{FC layers}(\text{flatten}(f)))$

There's no simple way to decompose this back to spatial locations because the FC transformation mixes information from all spatial positions in complex ways.

This is why CAM requires GAP architecture, and why Grad-CAM was developed to handle networks with FC layers by using gradient information instead.

Question 2. Feature Inversion and Network Understanding (22 marks)
The professor discussed: "Given the feature vector for X we can try to generate an image whose feature vector is similar to that image."

(a) Formulate the feature inversion optimization problem as the professor described. Explain why this is useful for understanding "what the CNN is doing" and "what level of detail it has kept and captured at different layers." (8 marks)

Answer: Feature inversion reconstructs images from feature representations by optimizing pixel values to match target activations, revealing what information is preserved at different network layers.

Feature Inversion Optimization Problem:

Given a target feature vector $\phi_l(x_0)$ from layer l of a CNN, find an image x^* such that:

$$x^* = \arg\min_x ||\phi_l(x) - \phi_l(x_0)||_2^2 + \lambda \mathcal{R}(x)$$

where:

- $\phi_l(x)$ is the feature representation of image x at layer l
- $\phi_l(x_0)$ is the target feature vector (from original image x_0)
- $\mathcal{R}(x)$ are regularization terms to ensure natural-looking images
- λ controls the regularization strength

Optimization Process: 1. Initialize x randomly (usually Gaussian noise) 2. Forward pass: compute $\phi_l(x)$ 3. Compute loss: $\mathcal{L} = ||\phi_l(x) - \phi_l(x_0)||_2^2$ 4. Backward pass: compute $\frac{\partial \mathcal{L}}{\partial x}$ 5. Update x using gradient descent: $x \leftarrow x - \alpha \frac{\partial \mathcal{L}}{\partial x}$ 6. Repeat until convergence

Understanding "What the CNN is Doing":

Feature inversion reveals:

(a) **Information Content:** What visual information is encoded in each layer's features

(b) **Invariances:** What transformations the network considers equivalent

(c) **Selectivity:** Which visual patterns activate specific neurons

(d) **Hierarchical Processing:** How representations change across layers

Layer-by-Layer Analysis:

By performing inversion at different layers, we understand:

- **Early Layers:** Preserve low-level details (edges, textures, colors)
- **Middle Layers:** Capture mid-level patterns (shapes, object parts)
- **Late Layers:** Focus on high-level semantic content, lose spatial details

This helps us understand the progressive abstraction process in CNNs.

(b) The professor showed reconstructions from different CNN layers. Explain his observation: "In the earlier layers information is redundant and we are able to capture the low-level details... when you go up in the network low-level details are lost." (8 marks)

Answer: Layer-wise reconstructions reveal a trade-off between low-level detail preservation and high-level semantic understanding as we progress through the network hierarchy.

Early Layer Reconstructions:

High Detail Preservation:

- Reconstructions are nearly pixel-perfect copies of originals
- Preserve fine textures, exact colors, and precise edge locations
- Can recover exact spatial arrangements and small-scale patterns
- Information content is high and redundant

Why Information is "Redundant": Early layers capture raw visual information with high fidelity:

- Adjacent pixels are highly correlated
- Local patterns repeat across the image
- Network hasn't yet performed significant dimensionality reduction
- Features still maintain direct correspondence to pixel values

Later Layer Reconstructions:

Loss of Low-Level Details:

- Reconstructions lose fine textures and exact colors
- Spatial locations become approximate rather than precise
- Background details often become blurred or missing
- Overall shape and semantic content preserved

Information Transformation Process:

As we progress through layers:

1. **Pooling Operations:** Reduce spatial resolution, losing precise location information 2. **Non-linear Activations:** Introduce information bottlenecks 3. **Increasing Receptive Fields:** Aggregate information over larger regions 4. **Semantic Specialization:** Neurons become selective for high-level concepts

Mathematical Perspective:

Information content decreases with depth:

$$I(\text{Image}; \phi_1) > I(\text{Image}; \phi_2) > ... > I(\text{Image}; \phi_L)$$

But task-relevant information is preserved:

$$I(\text{Class}; \phi_1) < I(\text{Class}; \phi_2) < ... < I(\text{Class}; \phi_L)$$

Practical Implications:

- **Transfer Learning:** Early layers transfer well because they capture universal visual features
- **Feature Extraction:** Late layers better for semantic tasks, early layers for texture/style tasks
- **Network Design:** Need to balance detail preservation with semantic understanding

(c) The professor mentioned regularization terms to make generated images "visually pleasing." List and explain the regularization terms he discussed for controlling "smoothness" and ensuring "neighboring pixels have similar intensities." (6 marks)

Answer: Regularization terms in feature inversion enforce natural image statistics to produce visually coherent reconstructions instead of noisy artifacts.

Total Variation (TV) Regularization:

Promotes spatial smoothness by penalizing large intensity differences between neighboring pixels:

$$\mathcal{R}_{TV}(x) = \sum_{i,j} \left[(x_{i+1,j} - x_{i,j})^2 + (x_{i,j+1} - x_{i,j})^2 \right]^{\beta/2}$$

where $\beta = 1$ (L1) or $\beta = 2$ (L2).

Effect: Encourages neighboring pixels to have similar intensities, reducing noise and creating smoother transitions.

L2 Pixel Norm Regularization:

Controls overall image magnitude:

$$\mathcal{R}_{L2}(x) = ||x||_2^2 = \sum_{i,j} x_{i,j}^2$$

Effect: Prevents pixel values from becoming arbitrarily large, keeping them in reasonable ranges.

Natural Image Prior:

Encourages reconstructions to follow natural image statistics:

$$\mathcal{R}_{\text{prior}}(x) = -\log P(x)$$

where $P(x)$ is a learned prior over natural images.

Alpha Norm Regularization:

Promotes sparsity in pixel values:

$$\mathcal{R}_{\alpha}(x) = \sum_{i,j} |x_{i,j}|^{\alpha}$$

where $\alpha < 2$ (commonly $\alpha = 6$).

Combined Objective:

The full optimization becomes:

$$\min_x ||\phi_l(x) - \phi_l(x_0)||_2^2 + \lambda_1 \mathcal{R}_{TV}(x) + \lambda_2 \mathcal{R}_{L2}(x) + \lambda_3 \mathcal{R}_\alpha(x)$$

Practical Effects:

- **Without regularization:** High-frequency noise, unnatural artifacts
- **With regularization:** Smooth, natural-looking images that preserve semantic content
- **Balance is crucial:** Too much regularization loses important details, too little creates artifacts

These regularization terms are essential because the feature matching objective alone can be satisfied by many unnatural images with high-frequency noise.

Question 3. Adversarial Attacks (23 marks)

Based on the professor's explanation: "We can generate a noise image that's called R with a very small magnitude such that when you add this noise image to the original image human eye is not able to visually distinguish this from the original."

(a) Formulate the adversarial attack optimization problem as described by the professor. Distinguish between "targeted attack" (where "we have a target class in mind") and "untargeted attack" (where "as long as the correct class is not predicted it doesn't matter"). (8 marks)

Answer: Adversarial attacks find minimal perturbations that fool neural networks while remaining imperceptible to humans, with different objectives for targeted versus untargeted scenarios.

General Adversarial Attack Formulation:

Given an original image x with true class y, find a perturbation r such that:

$$x' = x + r$$

where:

- $f(x') \neq y$ (misclassification occurs)
- $||r||_p \leq \epsilon$ (perturbation is small)
- x' is visually indistinguishable from x to humans

Untargeted Attack:

Goal: Make the network predict any class except the correct one.

$$\min_r ||r||_p \text{ subject to } f(x+r) \neq y \text{ and } x + r \in [0,1]^n$$

Alternative formulation (maximizing loss):

$$\max_r \mathcal{L}(f(x+r), y) \text{ subject to } ||r||_p \leq \epsilon$$

where $\mathcal{L}$ is the classification loss (e.g., cross-entropy).

Targeted Attack:

Goal: Make the network predict a specific target class $t \neq y$.

$$\min_r ||r||_p \text{ subject to } f(x+r) = t \text{ and } x + r \in [0,1]^n$$

Alternative formulation (minimizing target loss):

$$\min_r \mathcal{L}(f(x+r), t) \text{ subject to } ||r||_p \leq \epsilon$$

Key Differences:

- **Untargeted:** Easier to achieve, any misclassification is success
- **Targeted:** Harder to achieve, must fool network toward specific class
- **Untargeted:** Often requires smaller perturbations
- **Targeted:** May require larger perturbations to reach specific target

Common Attack Methods:

Fast Gradient Sign Method (FGSM):

$$r = \epsilon \cdot \text{sign}(\nabla_x \mathcal{L}(f(x), y))$$

Projected Gradient Descent (PGD): Iterative refinement with projection to constraint set.

The key insight is that small, carefully crafted perturbations can exploit the high-dimensional nature of neural network decision boundaries.

(b) Explain the difference between "white box approach" and "black box approach" for generating adversarial attacks. Describe the professor's explanation of using a "proxy network that mimics the predictions of the blackbox model." (8 marks)

Answer: White box attacks have full access to model internals while black box attacks only observe outputs, requiring different strategies like proxy networks for gradient estimation.

White Box Approach:

Available Information:

- Complete model architecture
- All learned parameters (weights and biases)
- Activation functions and layer types
- Access to gradients via backpropagation

Attack Strategy:

- Directly compute gradients: $\nabla_x \mathcal{L}(f(x), y)$
- Use gradient-based optimization methods (FGSM, PGD, CW)
- Can craft highly effective attacks with minimal perturbations
- Fast and computationally efficient

Mathematical Process:

$$r^{(t+1)} = \Pi_\epsilon \left(r^{(t)} + \alpha \cdot \text{sign}(\nabla_x \mathcal{L}(f(x + r^{(t)}), y)) \right)$$

where Π_ϵ projects to the constraint set.

Black Box Approach:

Limited Information:

- Only access to model outputs (predictions/probabilities)
- No knowledge of architecture or parameters
- Cannot compute gradients directly
- Must treat model as a "black box" function

Attack Strategies:

1. Query-Based Methods:

- Estimate gradients using finite differences
- $\frac{\partial f}{\partial x_i} \approx \frac{f(x+\delta e_i) - f(x-\delta e_i)}{2\delta}$
- Requires many queries to the target model
- Computationally expensive and detectable

2. Proxy Network Approach (Professor's Explanation):

Strategy: 1. Train a proxy network $g(x)$ to mimic the black box model $f(x)$ 2. Use the proxy for white box attacks: compute $\nabla_x \mathcal{L}(g(x), y)$ 3. Apply the adversarial examples to the original black box model 4. Rely on **transferability** of adversarial examples

Proxy Training Process:

- Collect input-output pairs from the black box: $\{(x_i, f(x_i))\}$
- Train proxy network: $\min_\theta \sum_i \mathcal{L}(g(x_i; \theta), f(x_i))$
- Use proxy for gradient computation

Transferability Principle: Adversarial examples often transfer between different models due to:

- Shared decision boundary properties
- Similar learned representations
- Common vulnerabilities in neural architectures

Advantages and Limitations:

White Box:

- **Pros:** Highly effective, computationally efficient
- **Cons:** Requires full model access (unrealistic in practice)

Black Box:

- **Pros:** Realistic attack scenario, works on deployed models
- **Cons:** Lower success rates, requires more computation or queries

The proxy approach is particularly clever because it transforms a black box problem into a white box problem using transferability.

(c) The professor discussed serious implications: autonomous cars could "misdetect a traffic sign" or "miss a crossing." Explain why adversarial robustness is challenging, including the professor's point about decision boundaries and the "trade-off between robustness and accuracy." (7 marks)

Answer: Adversarial robustness is fundamentally challenging due to the high-dimensional nature of neural network decision boundaries and the inherent trade-off between fitting training data accurately and being robust to perturbations.

High-Dimensional Decision Boundary Problem:

In high-dimensional input spaces (e.g., images with thousands of pixels):

- Decision boundaries have enormous surface area
- Most points are close to some decision boundary
- Small perturbations can easily cross boundaries
- Networks learn complex, non-linear decision surfaces

Mathematical Insight: For an n-dimensional input space, the volume near decision boundaries grows exponentially with n. This means most natural inputs are vulnerable to small adversarial perturbations.

Fundamental Trade-off: Robustness vs. Accuracy:

Accuracy Objective:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(f(x;\theta), y)]$$

This encourages the model to fit the training distribution precisely.

Robustness Objective:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\max_{||r||\leq\epsilon} \mathcal{L}(f(x+r;\theta), y) \right]$$

This requires the model to be correct even under worst-case perturbations.

Why They Conflict:

(a) **Overfitting to Training Data:** High accuracy models may rely on spurious correlations

(b) **Sharp Decision Boundaries:** Accurate models often have sharp boundaries that are easy to cross

(c) **Feature Brittleness:** Models may rely on features that are not robust to small changes

(d) **Capacity Limitations:** Finite model capacity must choose between fitting data and being robust

Safety-Critical Applications:

Autonomous Vehicles:

- **Stop Sign Attack:** Stickers on stop signs could cause misclassification as speed limit signs
- **Pedestrian Detection:** Adversarial patches could make pedestrians "invisible" to detection systems
- **Lane Detection:** Road markings could be adversarially modified

Medical Diagnosis:

- Small changes to medical images could flip diagnosis
- Could lead to missed diseases or false positives
- Life-or-death consequences from misclassification

Why Robustness is Fundamentally Hard:

1. Curse of Dimensionality: In high dimensions, "small" perturbations can be large in some directions.

2. Distribution Mismatch: Adversarial examples are outside the training distribution but within the input space.

3. Gradient-Based Learning: Standard training optimizes for average performance, not worst-case robustness.

4. Computational Constraints: Robust training requires solving expensive min-max optimization problems.

Current Approaches and Limitations:

- **Adversarial Training:** Include adversarial examples in training, but expensive and may overfit
- **Certified Defense:** Provide mathematical guarantees, but only for small perturbations
- **Detection Methods:** Try to detect adversarial examples, but can be circumvented

The fundamental challenge is that we want models that are simultaneously:

- Accurate on natural data
- Robust to all possible small perturbations
- Computationally efficient
- Generalizable to new domains

These objectives are often in tension, making adversarial robustness one of the most challenging problems in deep learning.

Question 4. AlexNet Architecture and Design (25 marks)
The professor described AlexNet as "the 2012 ImageNet winner" with "60 million parameters" that had to be "split into two branches" because "they were not able to fit the whole network into a single GPU."

(a) Analyze the AlexNet architecture as described by the professor. Explain why they used "a stride of four" in the first layer and why this "huge reduction in dimensionality" worked in early layers but "would lose performance if tried in the top of the network." (10 marks)

Answer: AlexNet's aggressive stride-4 in the first layer exploits the high redundancy in raw pixel data to reduce computational load, but such aggressive reduction would be harmful in later layers where information becomes more semantic and less redundant.

AlexNet Architecture Overview:

Layer Structure:

(a) Conv1: 11×11 kernels, stride=4, 96 filters
(b) MaxPool1: 3×3, stride=2
(c) Conv2: 5×5 kernels, stride=1, 256 filters
(d) MaxPool2: 3×3, stride=2
(e) Conv3: 3×3 kernels, stride=1, 384 filters
(f) Conv4: 3×3 kernels, stride=1, 384 filters
(g) Conv5: 3×3 kernels, stride=1, 256 filters
(h) MaxPool3: 3×3, stride=2
(i) FC1: 4096 units
(j) FC2: 4096 units
(k) FC3: 1000 units (ImageNet classes)

Why Stride=4 in First Layer Works:

1. High Pixel Redundancy: Natural images have high spatial correlation:

- Adjacent pixels are highly correlated (correlation coefficient ~ 0.95)

- Information is distributed redundantly across neighboring pixels
- Aggressive subsampling removes redundancy without losing essential information

2. Computational Efficiency: Input: $224 \times 224 \times 3 = 150,528$ values
After stride-4 conv: $55 \times 55 \times 96 = 290,400$ values

Computation reduction:

- Stride-1 would produce $214 \times 214 \times 96 = 4.4M$ values
- Stride-4 produces $55 \times 55 \times 96 = 0.29M$ values
- $15\times$ reduction in feature map size
- Massive reduction in subsequent computational cost

3. Effective Receptive Field: Large 11×11 kernels with stride-4 still capture sufficient local context while dramatically reducing spatial dimensions.

Why This Fails in Later Layers:

1. Information Becomes Semantic:

- Later layers encode high-level semantic features
- Each spatial location represents important object parts or relationships
- Aggressive pooling would destroy crucial semantic information

2. Reduced Redundancy:

- After several layers, redundancy is much lower
- Each feature map location contains unique, non-redundant information
- Pooling becomes information destructive rather than efficiency improving

3. Spatial Relationships Matter:

- Object part relationships become crucial for recognition
- Relative spatial positions encode important semantic information

- Aggressive reduction destroys these relationships

Mathematical Perspective:

Early layers: $H(X) \approx H(\text{Subsample}(X))$ (entropy preserved) **Late layers:** $H(X) \gg H(\text{Subsample}(X))$ (significant information loss)

This design principle - aggressive reduction early, conservative reduction late - became a fundamental CNN design pattern.

(b) The professor found it "striking" that AlexNet filters showed specialization: one GPU learned "more or less color insensitive" filters while the other learned "more color selective filters." Explain this phenomenon and what it suggests about "group convolution." (8 marks)

Answer: AlexNet's dual-GPU architecture accidentally discovered that splitting features into groups leads to specialized filter learning, laying the groundwork for modern group convolution techniques.

AlexNet's Dual-GPU Architecture:

Due to memory limitations (GPUs had only 3GB in 2012):

- Network split across two GPUs
- Each GPU handled half the filters (48 filters each in first layer)
- Limited communication between GPUs (only at certain layers)
- Forced independent learning in each branch

Observed Filter Specialization:

GPU 1 Filters:

- Learned mostly grayscale, color-insensitive filters
- Focused on edges, textures, and luminance patterns
- Similar to traditional computer vision filters (Gabor, edge detectors)
- More "generic" visual feature detection

GPU 2 Filters:

- Learned highly color-selective filters
- Specialized in color patterns and chromatic information

- Complementary to GPU 1's achromatic processing
- More specialized for color-based discrimination

Why This Specialization Occurred:

1. Limited Communication:

- Each GPU had to learn independently initially
- Forced diversification to avoid redundancy
- Each branch adapted to fill different functional roles

2. Complementary Learning:

- Network needed both luminance and chromatic information
- Division of labor emerged naturally
- Each GPU specialized where it could contribute most effectively

3. Gradient Flow Optimization:

- Each GPU received different gradient signals
- Specialized filters emerged to minimize overall network loss
- Natural feature differentiation occurred

Connection to Group Convolution:

Modern Group Convolution:

$$y_j = \sum_{i \in G_j} x_i * w_{i,j}$$

where G_j is the group of input channels connected to output channel j.

Benefits Suggested by AlexNet:

1. Feature Diversification:

- Groups force filters to learn different types of features
- Reduces redundancy in learned representations
- Increases overall network expressiveness

2. Computational Efficiency:

- Reduces parameters: full conv needs $C_{in} \times C_{out}$ connections
- Group conv needs $\frac{C_{in} \times C_{out}}{G}$ connections per group
- Maintains representational capacity with fewer parameters

3. Improved Generalization:

- Specialization prevents overfitting to specific features
- Forces network to learn more diverse, robust representations
- Similar to ensemble effects

Modern Applications:

ResNeXt: Uses group convolution as a design principle **MobileNet:** Extreme case (depthwise convolution) where groups = channels **ShuffleNet:** Combines group convolution with channel shuffling

Key Insight: AlexNet's accidental discovery showed that constraining connectivity can lead to more efficient and effective learning, inspiring an entire family of group-based architectures.

(c) List the key techniques AlexNet used to "mitigate overfitting" as mentioned by the professor, including their approach to "manually" decaying learning rate when "the loss was not improving." (7 marks)

Answer: AlexNet employed multiple overfitting mitigation strategies including data augmentation, dropout, weight decay, and manual learning rate scheduling based on loss plateau detection.

1. Data Augmentation:

Image Translations and Horizontal Flips:

- Random crops from 256×256 images to 224×224
- Horizontal reflections (50% probability)
- Created multiple versions of each training image
- Effective dataset size increase: $\sim 2048\times$ (multiple crops $\times$ flip)

PCA-based Color Augmentation:

- Principal Component Analysis on RGB pixel values
- Add multiples of found principal components to images

- Changes color intensity and hue while preserving object identity
- Helped network become invariant to lighting changes

2. Dropout Regularization:

Application:

- Applied to first two fully connected layers
- Dropout probability: 0.5
- Randomly set 50% of neurons to zero during training
- All neurons active during testing (with appropriate scaling)

Effect:

- Prevents co-adaptation of neurons
- Forces network to learn more robust representations
- Reduces overfitting to specific neuron combinations
- Acts like training an ensemble of networks

3. L2 Weight Decay:

Implementation:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{classification}} + \lambda \sum_i w_i^2$$

Parameters:

- Weight decay coefficient: $\lambda = 0.0005$
- Applied to all trainable parameters
- Penalizes large weights to prevent overfitting

4. Manual Learning Rate Scheduling:

Strategy:

- Initial learning rate: 0.01
- Monitor validation loss continuously

- When validation loss plateaus (stops improving), divide learning rate by 10
- Manual intervention by researchers watching training progress
- Process repeated until convergence

Rationale:

- High learning rate initially for fast convergence
- Lower learning rate when approaching minimum
- Prevents oscillation around optimal solution
- Allows fine-tuning of learned features

5. Early Stopping (Implicit):

Monitoring:

- Tracked both training and validation accuracy
- Stopped training when validation performance degraded
- Prevented excessive overfitting to training set

6. Architecture Choices for Regularization:

Local Response Normalization:

- Normalized responses across feature maps
- Implemented lateral inhibition mechanism
- Helped with generalization (though later found less effective)

Overlapping Pooling:

- Pool size 3×3 with stride 2 (overlapping)
- Slight regularization effect compared to non-overlapping
- Reduced top-1 error by 0.4% and top-5 error by 0.3%

Historical Context:

These techniques were pioneering in 2012:

- Dropout was a relatively new technique

- Data augmentation not yet standard practice
- Manual learning rate scheduling was common (automatic schedulers came later)
- Combination of techniques was novel and effective

The success of these regularization techniques in AlexNet established many best practices that are still used in modern deep learning.

Question 5. GoogleNet and Inception Modules (30 marks)
The professor explained GoogleNet's contributions: reducing parameters "from 60 million in AlexNet to 4 million" while achieving better performance.

(a) Explain the motivation for inception modules as described by the professor: "information can exist at multiple scales" and "if you use a fixed receptive field size at a layer actually you are restricting the type of information you can extract to only a fixed scale." (8 marks)

Answer: Inception modules address the fundamental limitation that fixed kernel sizes can only capture information at a single scale, while real-world objects and features exist at multiple scales simultaneously.

Multi-Scale Information Problem:

Real-World Visual Complexity:

- Objects appear at different sizes in images
- Same object type can have varying scales (near vs. far cars)
- Important features exist at multiple granularities
- Fine details and coarse structures both matter for recognition

Traditional CNN Limitation:

Each layer uses fixed kernel size (e.g., 3×3):

- Can only capture information at one spatial scale
- Receptive field grows layer by layer, but each layer is still scale-limited
- Forces network to choose between fine details and coarse patterns
- Suboptimal for objects that have both fine and coarse important features

Scale Examples in Vision:

Fine Scale ($1 \times 1, 3 \times 3$):

- Texture patterns, edge details
- Local color variations

- Small discriminative features

Medium Scale (5×5):

- Object parts (eyes, wheels, etc.)
- Medium-sized patterns
- Spatial relationships between local features

Coarse Scale (Pooling):

- Global shape information
- Spatial layout of major components
- Context and background relationships

Inception Solution - Multi-Scale Processing:

Parallel Processing: Instead of choosing one kernel size, use multiple kernel sizes in parallel:

- 1×1 convolutions for point-wise features
- 3×3 convolutions for local patterns
- 5×5 convolutions for larger patterns
- 3×3 max pooling for spatial reduction

Information Fusion:

- All scales processed simultaneously from same input
- Results concatenated to preserve all scale information
- Network learns to combine multi-scale features optimally
- No information loss from scale selection

Benefits:

1. Scale Invariance: Network can recognize objects regardless of their size in the image.

2. Feature Richness: Captures both fine-grained details and coarse structures simultaneously.

3. Adaptive Processing: Network learns which scales are most important for each class/feature.

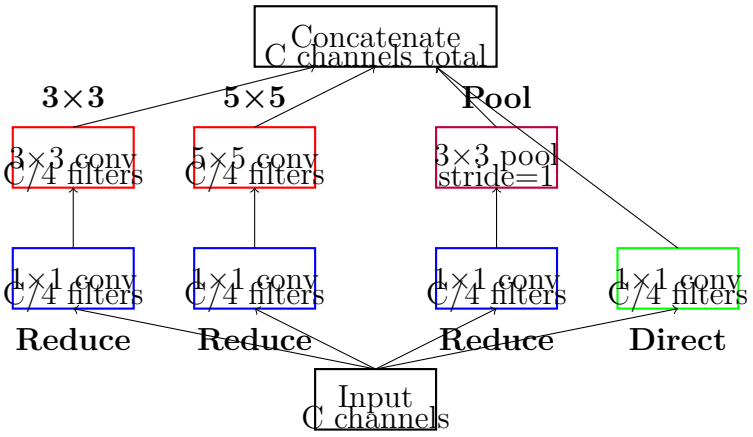
4. Efficiency: More information extracted per layer compared to single-scale approaches.

Biological Inspiration: Similar to human visual system:

- Multiple cell types with different receptive field sizes
- Parallel processing of different spatial frequencies
- Integration of multi-scale information for object recognition

This multi-scale approach became fundamental to modern CNN design, influencing architectures like ResNet, DenseNet, and many others.

(b) The professor described the "scalability problem" with naive inception modules: "the number of channels explodes." Draw the improved inception module and explain how "one by one convolution" solves this by ensuring "if the input layer has C channels the output layer after concatenation has C many channels as well." (12 marks)



Answer: The improved inception module uses 1×1 convolutions as "bottleneck" layers to reduce channel dimensions before expensive operations, controlling computational cost while maintaining multi-scale processing capability.

Scalability Problem in Naive Inception:

Naive Approach: Direct application of multiple kernel sizes to input with C channels:

- 1×1 conv: produces N_1 channels
- 3×3 conv: produces N_3 channels
- 5×5 conv: produces N_5 channels
- Pooling + conv: produces N_p channels
- **Total output:** $N_1 + N_3 + N_5 + N_p$ channels

Channel Explosion: If we want meaningful representation:

- Each branch needs sufficient channels (e.g., 128+ channels each)
- Output becomes $4 \times 128 = 512$ channels
- Next layer input is 512 channels
- Grows exponentially through network depth
- Leads to parameter explosion and computational infeasibility

Computational Cost Analysis:

For input $H \times W \times C$ and naive inception:

- 3×3 conv cost: $H \times W \times C \times N_3 \times 9$
- 5×5 conv cost: $H \times W \times C \times N_5 \times 25$
- Total cost grows as $O(C \times \text{total\_output\_channels})$

1×1 Convolution Solution:

Bottleneck Architecture:

(a) **Dimension Reduction:** Use 1×1 conv to reduce from C to $C/4$ channels

(b) **Multi-scale Processing:** Apply 3×3, 5×5 on reduced dimensions

(c) **Dimension Restoration:** Each branch outputs $C/4$ channels

(d) **Concatenation:** $4 \times (C/4) = C$ total output channels

How 1×1 Convolution Works:

Mathematical Operation: For input $x \in \mathbb{R}^{H \times W \times C}$ and 1×1 filter $w \in \mathbb{R}^{1 \times 1 \times C \times C'}$:

$$y_{i,j,k} = \sum_{c=1}^{C} w_{1,1,c,k} \cdot x_{i,j,c} + b_k$$

Effect:

- Acts as learned linear combination across channels
- Reduces/increases channel dimension without affecting spatial dimensions
- Introduces non-linearity through ReLU activation
- Very efficient: only $C \times C'$ parameters per filter

Computational Savings:

Without Bottleneck (Naive):

- 5×5 conv: $H \times W \times C \times (C/4) \times 25 = 6.25 \times HWC^2/4$
- Very expensive for large C

With Bottleneck:

- 1×1 reduction: $H \times W \times C \times (C/4) = HWC^2/4$
- 5×5 conv: $H \times W \times (C/4) \times (C/4) \times 25 = 25HWC^2/16$
- Total: $HWC^2/4 + 25HWC^2/16 = HWC^2(4 + 25)/16 = 1.8 \times HWC^2$
- **Speedup:** $\frac{6.25}{1.8} \approx 3.5\times$ faster

Key Benefits:

1. Controlled Growth: Output channels = Input channels, preventing exponential growth

2. Computational Efficiency: Dramatic reduction in FLOPs while maintaining representational capacity

3. Feature Learning: 1×1 convolutions learn optimal channel combinations

4. Scalability: Can stack many inception modules without parameter explosion

This bottleneck design became a fundamental principle in efficient CNN architectures, used in ResNet, MobileNet, and many others.

(c) Explain GoogleNet's second contribution: "adding fully connected layers into the intermediate layers of the network." Discuss both benefits (gradient flow, multi-scale recognition) and drawbacks (reproducibility issues, loss of generalization) as mentioned by the professor. (10 marks)

Answer: GoogleNet's auxiliary classifiers provide gradient flow improvements and multi-scale supervision but introduce training complexity and potential overfitting issues that limit their practical adoption.

Auxiliary Classifier Architecture:

Placement:

- Two auxiliary classifiers added at intermediate layers
- Located after inception modules 4a and 4d
- Each consists of: AvgPool → 1×1 Conv → FC → FC → Softmax
- Same 1000-class outputs as main classifier

Training Procedure:

- Total loss = Main loss + 0.3 × Aux1 loss + 0.3 × Aux2 loss
- All three classifiers trained simultaneously
- Auxiliary classifiers discarded during inference
- Only main classifier used for actual predictions

Benefits:

1. Improved Gradient Flow:

Vanishing Gradient Problem:

- GoogleNet has 22 layers - very deep for 2014
- Gradients become very small when backpropagated through many layers
- Early layers receive weak training signals

Auxiliary Classifier Solution:

- Provides direct gradient paths to intermediate layers
- Shorter backpropagation paths: gradients don't have to travel through all 22 layers
- Stronger training signals reach early and middle layers
- Helps combat vanishing gradients before ResNet's skip connections

Mathematical Insight: Gradient magnitude at layer ℓ: $||\nabla_{\theta_\ell}\mathcal{L}|| \propto \prod_{i=\ell}^{L} ||\nabla a_i||$

Auxiliary classifiers provide additional terms that don't require the full product.

2. Multi-Scale Feature Learning:

Intermediate Supervision:

- Forces intermediate layers to learn discriminative features
- Prevents early layers from becoming simple feature detectors
- Ensures useful representations at multiple network depths
- Creates multiple levels of abstraction for classification

Feature Quality:

- Middle layers must be good enough for classification
- Improves transferability of intermediate features
- Creates richer feature hierarchies

3. Regularization Effect:

- Acts as implicit regularization on intermediate features
- Prevents overfitting to specific feature combinations
- Encourages robust feature learning throughout network

Drawbacks:

1. Reproducibility Issues:

Training Instability:

- Three different loss terms with different dynamics
- Auxiliary loss weights (0.3) were manually tuned
- Sensitive to hyperparameter choices
- Different runs could converge to different solutions

Implementation Complexity:

- More complex training procedure
- Requires careful balancing of loss terms
- Harder to reproduce results across different frameworks
- Additional architectural components to implement

2. Loss of Generalization:

Overfitting Risk:

- Auxiliary classifiers might overfit to training data
- Intermediate features become biased toward auxiliary tasks
- May hurt performance on transfer learning tasks
- Creates task-specific representations too early in network

Feature Specialization:

- Intermediate layers forced to be classifiers rather than feature extractors
- Reduces flexibility of learned representations
- May prevent learning of more general, transferable features

3. Limited Practical Impact:

Empirical Results:

- Improvement was modest (1-2% accuracy gain)
- Benefits didn't consistently transfer to other tasks
- ResNet's skip connections provided better gradient flow solution
- Most practitioners found auxiliary classifiers not worth the complexity

Historical Impact:

Lessons Learned:

- Highlighted importance of gradient flow in deep networks
- Inspired better solutions (skip connections in ResNet)
- Showed that intermediate supervision can help but at a cost
- Demonstrated need for simpler, more principled approaches

Modern Perspective: Auxiliary classifiers are rarely used today because:

- ResNet skip connections solve gradient flow more elegantly
- Batch normalization helps with training stability
- Better optimizers (Adam, etc.) handle deep networks better
- Simpler architectures achieve better results

GoogleNet's auxiliary classifiers were innovative but ultimately a stepping stone to better solutions for training very deep networks.

Question 6. ResNet and Skip Connections (25 marks)

The professor explained the key insight: "After a certain number of layers performance doesn't improve... it actually becomes even worse" and ResNet's solution using skip connections.

(a) Analyze the professor's explanation of why deeper networks perform worse: "when you multiply all those gradients through all of these layers you don't get useful gradients to the earlier parts of the network." Explain why this isn't simply overfitting. (8 marks)

Answer: The degradation problem in deep networks is fundamentally different from overfitting - it's a training optimization problem where even training accuracy degrades, caused by vanishing gradients and optimization difficulties rather than generalization failure.

Empirical Observation - The Degradation Problem:

Experimental Evidence:

- 20-layer network: 91.25% training accuracy, 89.43% test accuracy
- 56-layer network: 90.16% training accuracy, 87.23% test accuracy
- Deeper network performs worse on **both** training and test sets
- This contradicts typical overfitting patterns

Why It's Not Overfitting:

Overfitting Characteristics:

- Training accuracy increases while test accuracy decreases
- Model memorizes training data but fails to generalize
- More parameters → better training fit, worse test performance
- Training loss continues to decrease

Degradation Problem Characteristics:

- **Both** training and test accuracy degrade
- Training loss stops decreasing or even increases
- More parameters → worse performance on all metrics

- Clear optimization failure, not generalization failure

Vanishing Gradient Analysis:

Mathematical Foundation: For a deep network with L layers, the gradient at layer ℓ is:

$$\frac{\partial \mathcal{L}}{\partial \theta_\ell} = \frac{\partial \mathcal{L}}{\partial a_L} \prod_{i=\ell+1}^{L} \frac{\partial a_i}{\partial a_{i-1}} \frac{\partial a_\ell}{\partial \theta_\ell}$$

Chain Rule Problem: The gradient involves a product of $L - \ell$ terms. Each term $\frac{\partial a_i}{\partial a_{i-1}}$ can be:

- Small (< 1): Product approaches zero exponentially
- Large (> 1): Product explodes exponentially
- Both cases make training unstable

Activation Function Impact:

Sigmoid/Tanh:

- Derivative range: $(0, 0.25]$ for sigmoid
- Product of many small terms $\rightarrow$ vanishing gradients
- Early layers receive virtually no training signal

ReLU:

- Derivative: 1 (if $x > 0$) or 0 (if $x \leq 0$)
- Helps with vanishing gradients but creates "dying ReLU" problem
- Still suffers from optimization difficulties in very deep networks

Optimization Landscape Perspective:

Loss Surface Complexity:

- Deeper networks have more complex loss landscapes
- More local minima and saddle points
- Harder for optimizers to find good solutions
- Path dependence in optimization trajectory

Identity Mapping Problem: Deep networks should theoretically perform at least as well as shallow ones:

- Extra layers could learn identity mappings
- In practice, learning identity mappings is difficult
- Networks struggle to approximate $H(x) = x$ when needed
- This suggests optimization rather than representational limitations

Empirical Evidence Against Overfitting:

1. Training Performance Degradation: If it were overfitting, training accuracy should still be high or increasing.

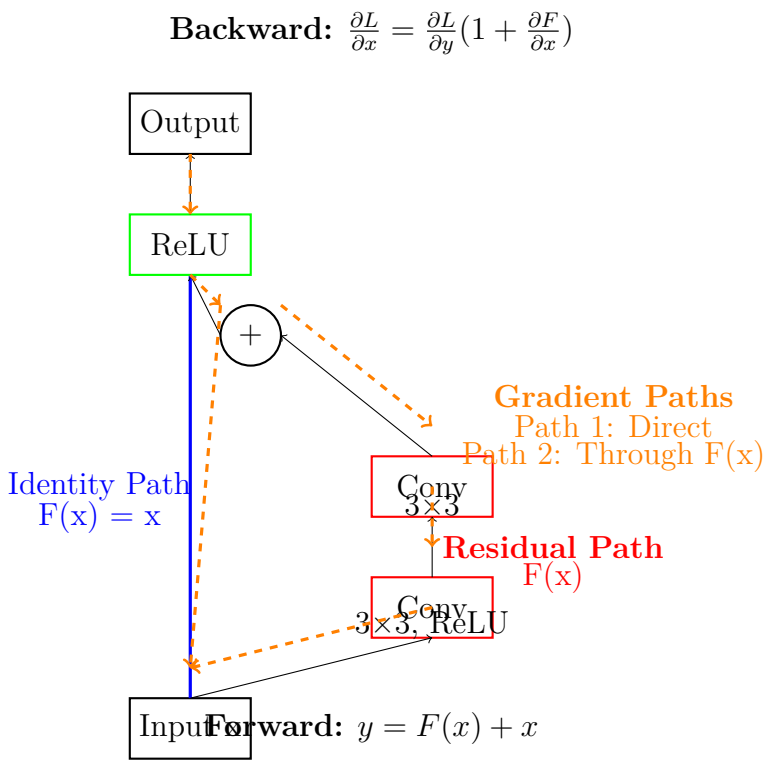
2. Consistent Across Datasets: The problem appears on various datasets, not just specific overfitting scenarios.

3. Architecture Independence: Different architectures show similar degradation patterns.

4. Regularization Doesn't Help: Standard overfitting solutions (dropout, weight decay) don't solve the degradation problem.

This analysis led to the insight that the fundamental problem is optimization difficulty, not overfitting, which motivated the skip connection solution in ResNet.

(b) Draw a ResNet block and explain the professor's description: "in forward pass we have a function of the input as well as the identity being propagated" and "in backward pass gradient will flow through two paths." (10 marks)



Answer: ResNet blocks combine a residual function F(x) with an identity skip connection, enabling gradient flow through both transformed and direct paths during backpropagation.

ResNet Block Architecture:

Forward Pass Components:

(a) **Input:** x (from previous layer)

(b) **Residual Function:** $F(x)$ (typically 2-3 conv layers)

(c) **Skip Connection:** Direct path carrying x

(d) **Addition:** Element-wise sum $F(x) + x$

(e) **Output:** $y = F(x) + x$ (usually followed by ReLU)

Forward Pass Mathematics:

Traditional Deep Network:

$$y = H(x)$$

Network must learn the full mapping $H(x)$ from scratch.

ResNet Formulation:

$$y = F(x) + x$$

where:

- $F(x)$ is the residual function (learned by conv layers)
- x is the identity mapping (skip connection)
- Network only needs to learn the residual $F(x) = H(x) - x$

Key Insight: Learning $F(x) = H(x) - x$ is easier than learning $H(x)$ directly, especially when $H(x) \approx x$ (near-identity mappings).

Backward Pass - Dual Gradient Paths:

Gradient Computation: Using chain rule on $y = F(x) + x$:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y}\frac{\partial y}{\partial x} = \frac{\partial \mathcal{L}}{\partial y}\left(\frac{\partial F(x)}{\partial x} + \frac{\partial x}{\partial x}\right) = \frac{\partial \mathcal{L}}{\partial y}\left(\frac{\partial F(x)}{\partial x} + 1\right)$$

Two Gradient Paths:

Path 1 - Direct (Identity):

- Contribution: $\frac{\partial \mathcal{L}}{\partial y} \cdot 1$
- Always present regardless of $F(x)$
- Provides unimpeded gradient flow
- Prevents vanishing gradients

Path 2 - Residual:

- Contribution: $\frac{\partial \mathcal{L}}{\partial y} \cdot \frac{\partial F(x)}{\partial x}$
- Depends on learned residual function
- Can vanish if $F(x)$ saturates
- Provides additional learning signal

Benefits of Dual Path Design:

1. Gradient Flow Guarantee: Even if $\frac{\partial F(x)}{\partial x} \to 0$, gradients still flow through identity path:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y}(0 + 1) = \frac{\partial \mathcal{L}}{\partial y}$$

2. Optimization Advantage:

- If optimal mapping is identity: $F(x) \to 0$ (easier than learning $H(x) = x$)
- If optimal mapping is complex: $F(x)$ learns the residual
- Network can adaptively choose between identity and transformation

3. Multi-Scale Learning:

- Identity path preserves low-level features
- Residual path learns new transformations
- Combination enables both preservation and transformation

Training Dynamics:

Early Training:

- $F(x)$ starts near zero (due to initialization)
- Output $y \approx x$ (near-identity)
- Strong gradient flow enables early layer training

During Training:

- $F(x)$ gradually learns useful residuals
- Identity path maintains stable gradient flow
- Network learns both to preserve and transform features

Theoretical Foundation:

Highway Networks Connection: ResNet can be viewed as a special case of highway networks with:

- Transform gate: $T = 1$ (always transform)

- Carry gate: $C = 1$ (always carry)
- Fixed gating removes learning overhead

This dual-path design revolutionized deep learning by solving the degradation problem and enabling training of networks with hundreds of layers.

(c) The professor mentioned two interpretations of why ResNet works: the "smoother loss function" hypothesis and viewing ResNet as "an ensemble of multiple sub networks." Explain both interpretations and discuss why "there is no theorem showing this." (7 marks)

Answer: ResNet's success has multiple theoretical explanations including loss landscape smoothing and ensemble-like behavior, but the lack of formal proofs reflects the general challenge of theoretically understanding deep neural networks.

Interpretation 1: Smoother Loss Function Hypothesis

Core Idea: Skip connections fundamentally change the loss landscape, making it smoother and easier to optimize.

Mathematical Intuition:

Traditional Network Loss Surface:

- Highly non-convex with many local minima
- Sharp valleys and plateaus
- Gradient descent can get trapped in poor local optima
- Small parameter changes can cause large loss changes

ResNet Loss Surface:

- Skip connections provide "shortcuts" across the loss landscape
- Smoother gradients due to identity path contribution
- Fewer sharp local minima
- More predictable optimization behavior

Empirical Evidence:

- Loss visualization studies show smoother landscapes for ResNets

- More consistent convergence across different initializations
- Less sensitivity to learning rate choices
- Better optimization even with simple SGD

Gradient Smoothing Effect: The dual gradient paths create more stable gradient flow:

$$\frac{\partial \mathcal{L}}{\partial x} = \frac{\partial \mathcal{L}}{\partial y}\left(1 + \frac{\partial F(x)}{\partial x}\right)$$

The "+1" term ensures gradients never completely vanish, providing consistent optimization signals.

Interpretation 2: Ensemble of Sub-Networks

Core Idea: ResNet can be viewed as an exponential ensemble of sub-networks of different depths, each taking different paths through the skip connections.

Mathematical Framework:

For a ResNet with n blocks, each with a skip connection, there are 2^n possible paths:

- Each block can either apply transformation $F_i(x)$ or skip it
- Path through blocks 1,3,5: $x \to F_1(x) \to F_3(F_1(x)) \to F_5(F_3(F_1(x)))$
- Total computation is sum over all paths weighted by their contributions

Unraveled View: A ResNet can be written as:

$$y = \sum_{\text{paths } p} \prod_{i \in p} F_i(x)$$

where the sum is over all possible paths through the network.

Ensemble Properties:

1. Implicit Averaging:

- Different paths contribute different "votes" to final prediction
- Naturally provides ensemble-like robustness

- Multiple depths provide multiple levels of abstraction

2. Graceful Degradation:

- If some blocks fail or perform poorly, other paths compensate
- Network is robust to individual component failures
- Training naturally balances path contributions

3. Multi-Scale Processing:

- Short paths provide coarse features
- Long paths provide detailed transformations
- Ensemble captures features at multiple scales

Empirical Support:

Path Analysis Studies:

- Gradient flow primarily through shorter paths during training
- Longer paths contribute less but still matter for performance
- Removing random subsets of blocks degrades performance gracefully
- Path importance follows specific distribution patterns

Why "There is No Theorem Showing This":

1. Complexity of Deep Networks:

Mathematical Challenges:

- Deep networks are highly non-linear systems
- Interactions between layers are complex and input-dependent
- Optimization dynamics involve stochastic processes (SGD, mini-batches)
- High-dimensional parameter spaces resist theoretical analysis

2. Multiple Concurrent Effects:

- Skip connections affect optimization, generalization, and representation simultaneously

- Hard to isolate individual causal mechanisms
- Different explanations may all be partially correct
- Interactions between effects complicate theoretical analysis

3. Empirical vs. Theoretical Understanding:

Deep Learning Theory Gap:

- Most successful deep learning innovations discovered empirically
- Theory often lags behind practical breakthroughs
- Theoretical tools are still developing for deep networks
- Phenomena may be too complex for current mathematical frameworks

4. Statistical vs. Deterministic Analysis:

- Deep learning involves high-dimensional statistics
- Traditional optimization theory assumes deterministic settings
- Real networks trained with noisy, finite data
- Generalization requires understanding of statistical learning theory

Current State:

Partial Results:

- Some theoretical insights for specific cases (linear networks, infinite width)
- Empirical studies support various hypotheses
- No unified theory explaining all aspects of ResNet success
- Active area of ongoing research

Practical Implications:

- Architecture design remains largely empirical
- Multiple theoretical perspectives provide useful intuitions
- Importance of ablation studies and empirical validation
- Need for continued theoretical research in deep learning