

Deep Learning Winter 2025–2026

Homework 4 Solutions

Mahmoud Abade (206773756)

Firas Dwere (214225021)

January 28, 2026

Question 3: Representation Geometry and Learning Objectives

(a) Distributional representations (7 points)

Two words that never appear together in a corpus can still have similar embeddings because word embeddings are learned based on the **distributional hypothesis** – “words that occur in similar contexts tend to have similar meanings.”

Here’s why this works:

Words don’t need to co-occur directly with each other to be similar. Instead, they need to appear in similar contexts. For example, consider the words “cat” and “dog”:

- “cat” might appear with: “pet”, “furry”, “meow”, “small”
- “dog” might appear with: “pet”, “furry”, “bark”, “small”

Even if “cat” and “dog” never appear in the same sentence together, they both share many context words (“pet”, “furry”, “small”). The embedding model learns to place words in similar positions in the vector space if they have similar context distributions.

This is the core idea behind algorithms like Word2Vec and GloVe. They create embeddings where:

$$\text{similarity}(w_1, w_2) \approx \text{similarity}(\text{contexts}(w_1), \text{contexts}(w_2))$$

So basically, if two words can be substituted in similar sentences (even if they never actually appear together), they’ll end up with similar embeddings.

(b) Geometry of embedding spaces (7 points)

Why anisotropy is problematic:

Anisotropy means embedding vectors cluster in a narrow cone rather than spreading uniformly across the space. This causes problems for similarity-based reasoning because:

1. **High baseline similarity:** When most vectors point in similar directions, even unrelated words will have relatively high cosine similarity. This makes it hard to distinguish truly similar words from random pairs.

2. **Reduced discriminative power:** The embedding space isn't being used efficiently – we have a high-dimensional space but vectors only occupy a small region of it, wasting representational capacity.
3. **Biased similarity judgments:** Similarity scores get compressed into a narrow range, making it harder to rank or threshold effectively.

High-level approach to mitigate anisotropy:

One common approach is **post-processing normalization**, specifically:

All-but-the-Top: Remove the dominant principal components from the embeddings. This works by:

1. Computing the principal components (PCA) of the embedding matrix
2. Removing the top k components (typically 1-5) that capture the most variance
3. This "spreads out" the embeddings more uniformly across the space

Another approach is to use **better training objectives** that explicitly encourage isotropy, like adding regularization terms that penalize high correlation between embedding dimensions or using contrastive learning with hard negative sampling.

(c) Contrastive learning and representation structure (6 points)

How contrastive learning shapes embedding geometry:

Contrastive learning uses the objective:

$$\mathcal{L} = -\log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\sum_j \exp(\text{sim}(z_i, z_j)/\tau)}$$

This pulls positive pairs closer and pushes negative pairs apart. The geometry that emerges has several key properties:

1. **Clusters by semantic similarity:** Similar items form tight clusters, while dissimilar items are pushed far apart on the hypersphere (when using cosine similarity).
2. **Uniform distribution on hypersphere:** Good contrastive learning tends to spread representations uniformly on the unit sphere, avoiding the anisotropy problem.

Why relative rather than absolute representations:

Contrastive learning encourages *relative* representations because:

- The loss only cares about *relative distances* between examples, not absolute positions
- What matters is whether $\text{sim}(x, x^+) > \text{sim}(x, x^-)$, not the actual values
- This makes the model focus on "this is more similar to that than to those" rather than learning specific coordinate values

Why they transfer well:

Contrastive representations transfer well to downstream tasks because:

1. **General similarity structure:** By learning to distinguish between different data variations, the model captures general semantic structure that's useful for many tasks.
2. **Invariance to task-irrelevant features:** The model learns to ignore irrelevant variations (e.g., color jitter, cropping) and focus on semantic content.
3. **No task-specific biases:** Since no labels are used during pretraining, the representations aren't biased toward any particular downstream task – they learn general-purpose features.
4. **Rich feature hierarchy:** The need to distinguish many negative examples forces the model to learn detailed, discriminative features that are useful across tasks.

Question 4: Transformers, Vision Transformers, and Generalization

(a) Depth and abstraction in Transformers (7 points)

Even though all Transformer layers have the same structure (self-attention + FFN), stacking them leads to increasingly abstract representations through **compositional processing**:

How abstraction emerges across layers:

1. Early layers - Low-level patterns:

- Attention patterns focus on local, syntactic relationships
- Features capture simple patterns (e.g., adjacent words, simple phrases)
- Example: "the cat" – just learning that "the" and "cat" often appear together

2. Middle layers - Compositional semantics:

- Attention can now attend to the refined features from earlier layers
- Can capture more complex relationships by composing lower-level patterns
- Example: Understanding "the cat that sat on the mat" as a complete noun phrase

3. Deep layers - Abstract concepts:

- Attention operates on already-processed semantic representations
- Can capture long-range dependencies and abstract relationships
- Example: Resolving pronouns, understanding document-level themes

Key mechanism - Composition through attention:

Each layer can attend to the outputs of previous layers. So layer L doesn't just see the input – it sees the *processed representations* from layer $L - 1$. This allows:

$$\text{Layer } L : h^{(L)} = \text{Attention}\left(\underbrace{h^{(L-1)}}_{\text{already processed}}\right) + \text{FFN}(h^{(L-1)})$$

By repeatedly applying this, features become progressively more abstract. It's like building a hierarchy: simple edges \rightarrow shapes \rightarrow objects \rightarrow scenes.

(b) Inductive bias: CNNs vs. Vision Transformers (7 points)

What is inductive bias?

Inductive bias refers to the assumptions built into a model's architecture that guide it toward certain types of solutions. It's the "prior knowledge" encoded in how the model is structured, which helps it learn more efficiently on certain tasks.

Key inductive bias in CNNs that's absent in ViT:

Translation equivariance and locality bias

CNNs have strong built-in assumptions:

- **Locality:** Convolutions only look at local neighborhoods, assuming nearby pixels are more related than distant ones
- **Translation equivariance:** The same features are detected regardless of where they appear in the image (weight sharing across spatial locations)
- **Spatial hierarchy:** Progressive downsampling builds a spatial hierarchy automatically

Vision Transformers, on the other hand:

- Treat the image as a sequence of patches with *no inherent spatial structure*
- Self-attention can attend to any patch regardless of distance (global receptive field from layer 1)
- Must *learn* spatial relationships from data rather than having them built-in

Consequence on small datasets like Flickr8k:

On small datasets, CNNs typically outperform Vision Transformers because:

1. **CNNs need less data:** The locality and translation equivariance biases are correct for natural images, so CNNs get a "head start" – they don't need to learn these properties from scratch.
2. **ViTs need to learn everything:** Without built-in spatial biases, ViTs must learn from data that:
 - Nearby pixels are related
 - The same object looks similar when shifted
 - Edges and textures are local phenomena
3. **Risk of overfitting:** ViTs have more parameters and more flexibility. On small datasets like Flickr8k (only 8,000 images), this flexibility becomes a curse – the model can memorize training data rather than learning generalizable features.
4. **Sample efficiency:** CNNs make better use of limited data because their inductive biases align with the structure of visual data.

This is why ViTs typically need either large datasets (like ImageNet) or strong pre-training to match or exceed CNN performance.

(c) Self-supervised pretraining and data efficiency (6 points)

Why self-supervised pretraining improves generalization:

Self-supervised learning (like masked image modeling) helps even without labels because:

1. **Learns general visual representations:**
 - By reconstructing masked patches, the model learns about textures, shapes, objects, and their spatial relationships

- These features are general-purpose – useful for many downstream tasks, not just reconstruction

2. **Better initialization:**

- Instead of starting from random weights, the model starts with weights that already encode meaningful visual features
- This is especially important for ViTs, which otherwise need lots of data to learn spatial biases

3. **Reduces reliance on labeled data:**

- The model can learn from unlimited unlabeled images
- Pretraining on large unlabeled datasets provides features that generalize better than training only on small labeled sets

Relation to zero-shot performance:

For vision-language tasks (like in this homework):

1. **Rich visual features:** Self-supervised pretraining creates visual representations that capture semantic content, not just low-level features.
2. **Better alignment:** When we later align vision and language, we're aligning *meaningful visual features* with text, not random features. This makes the alignment more effective.
3. **Generalization through transfer:** The visual features learned during pretraining transfer to the vision-language task. The model doesn't need to learn "what is a cat" from the limited caption data – it already learned this during self-supervised pretraining.
4. **Zero-shot capability:** Good pretrained representations capture semantic similarities. So even for objects/concepts not seen during vision-language training, the model can:
 - Recognize visual features from pretraining
 - Match them to similar text descriptions
 - Generalize to new categories based on learned similarities

Basically, self-supervised pretraining gives the ViT a strong visual foundation, so the vision-language alignment phase only needs to learn the mapping between vision and language, not learn visual understanding from scratch. This dramatically improves data efficiency and zero-shot performance.