Transformer iN Transformer Research Paper Notes

---

1. Transformer iN Transformer (TNT) Architecture

Summary

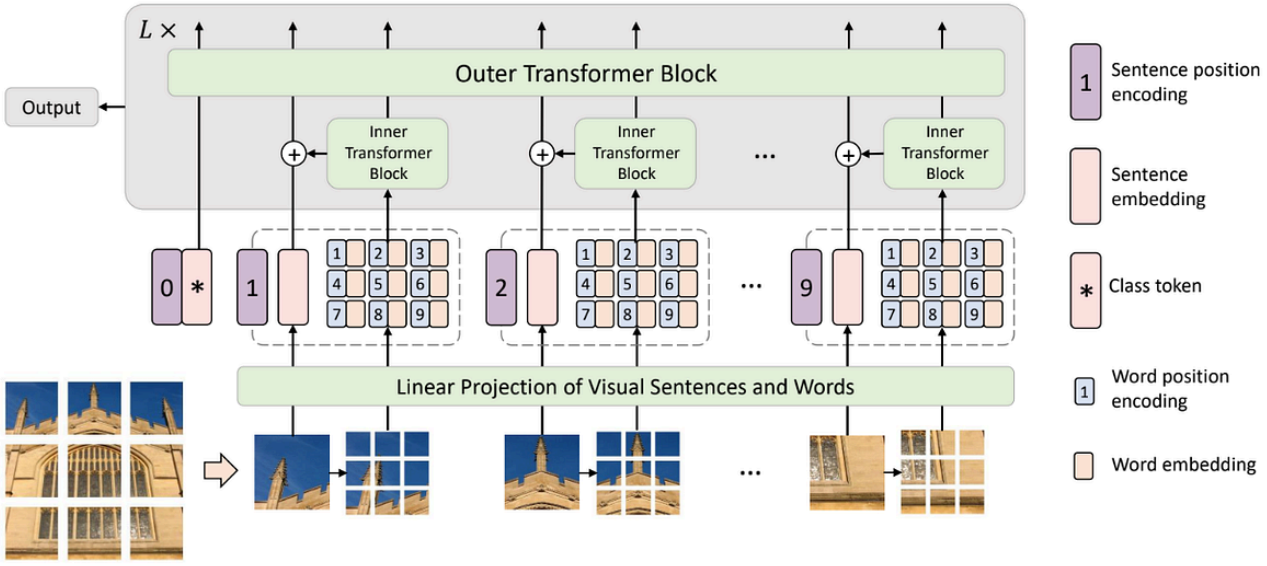
TNT addresses the loss of fine detail in standard Vision Transformers by using a nested architecture that processes small "visual words" (local details) within larger "visual sentences" (global context) to capture both specific features and the overall image structure.

Key Concepts

- **The "Granularity" Problem:** Standard Vision Transformers (like ViT) divide an image into large squares (patches). When these patches are processed, the model treats the

patch as a single "blob" of information, ignoring the specific arrangement of pixels inside that patch.

- **Visual Sentences & Visual Words:** TNT breaks images down hierarchically:
  - **Visual Sentence:** A larger patch of the image (e.g., 16x16 pixels).
  - **Visual Word:** A smaller sub-patch inside the sentence (e.g., 4x4 pixels).
  - **Analogy:** Think of the image as a **Paragraph**. The distinct objects or regions are **Sentences**, and the tiny details (textures, edges) making up those objects are the **Words**.
- **Dual Transformer Blocks:** The system uses two processing engines:
  - **Inner Transformer (Tin):** Focuses on the "words" to understand local details.
  - **Outer Transformer (Tout):** Focuses on the "sentences" to understand the global picture.



How it Works

The TNT architecture functions by embedding a smaller transformer inside a larger one to ensure no detail is lost.

Explore

1. **Hierarchical Division:** The input image is first split into large patches ("Visual Sentences"). Immediately, these sentences are subdivided into smaller sub-patches ("Visual Words").
2. **Position Encoding:** The model tags the data with location information so it knows where things are:
   - **Sentence Position:** Where the patch is in the overall image.
   - **Word Position:** Where the sub-patch is inside the main patch.
3. **Inner Processing (Tin):** The **Inner Transformer** looks at the "Visual Words" independently. It calculates how these small details relate to one another (e.g., how the

curve of an eye relates to the eyelid) to extract fine-grained features.

4. **Feature Aggregation:** The detailed features found in the "words" are combined (projected) to match the size of the "sentence" data. These details are then added to the main sentence embedding.
5. **Outer Processing (Tout):** The **Outer Transformer** processes the enhanced "Visual Sentences" (now rich with local details) to understand the relationships between different parts of the image (e.g., how the head relates to the body).

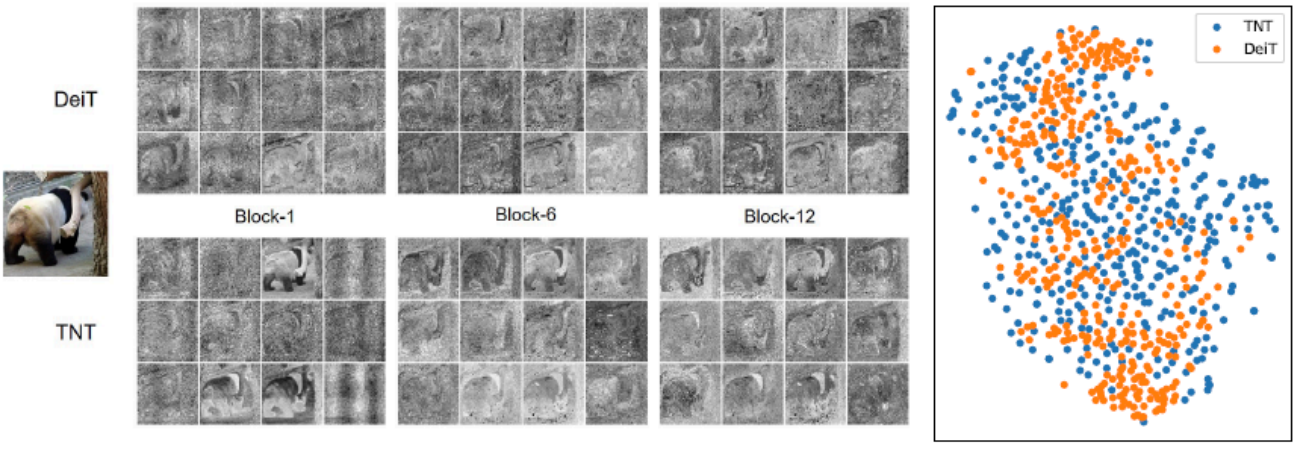
Conclusion

- **Preservation of Structure:** Unlike standard models that "corrupt" local data by flattening patches too early, TNT successfully preserves the internal structure and pixel relationships within patches.
- **Efficiency:** Despite having two transformer layers, the computational cost (FLOPs) and parameter count increase is negligible because the inner network weights are shared across all visual sentences.
- **Outcome:** The model achieves a better balance between accuracy and complexity, providing a more robust solution for visual recognition tasks that require noticing fine details.

---

2. Two Levels of Granularity & Architecture

TNT borrows terminology from Natural Language Processing to split images into coarse "Visual Sentences" (patches) and fine "Visual Words" (sub-patches), using a dual-transformer system to process local details and global structure simultaneously.



(a) Feature maps in Block-1/6/12. (b) T-SNE of Block-12.

Key Concepts & Definitions

- **Hierarchical Granularity:** The core innovation of TNT is dividing the image into two distinct levels of detail, rather than treating it as one flat grid.
  - **Level 1: Visual Sentences (Global):** These are standard large image patches (e.g., 16×16 pixels). They represent the "coarse" view of the image.
  - **Level 2: Visual Words (Local):** Each sentence is further divided into smaller sub-patches (e.g., 4×4 pixels). These represent the "fine" view.

> - **The Goal:** The model understands that the "eye" (word) belongs to the "face" (sentence), which belongs to the "group" (image).

- **Dual-Transformer Mechanism:** To handle these two levels, TNT uses two specific engines:
  - **Inner Transformer (T_{in}):** Processes the "words" to extract local features.
  - **Outer Transformer (T_{out}):** Processes the "sentences" to understand global relationships.

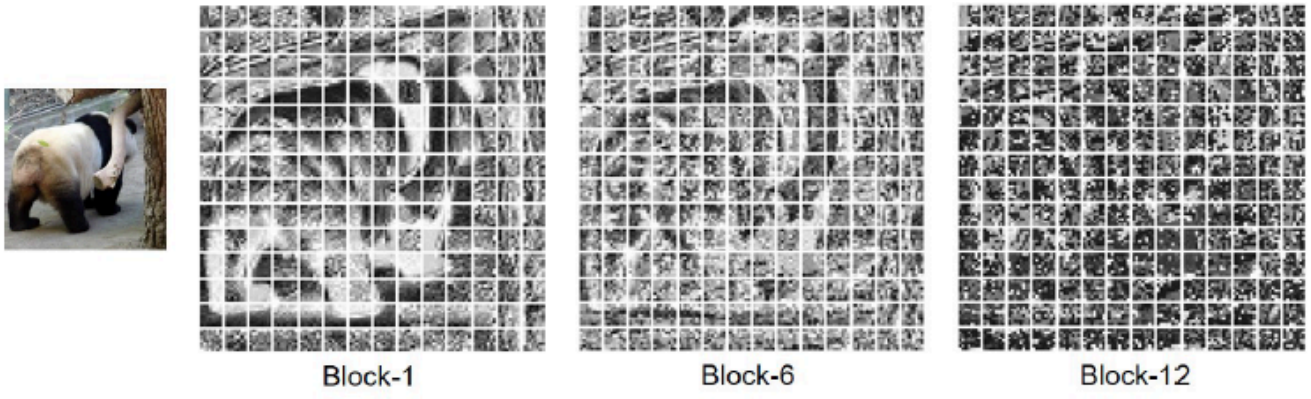
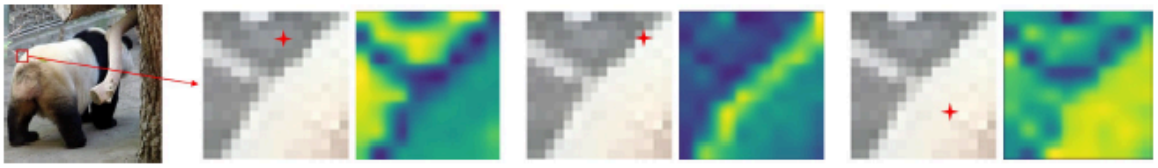


Figure 4: Visualization of the averaged word embeddings of TNT-S.



How it Works

The TNT architecture processes data through a specific flow to integrate these two levels of granularity.

1. Splitting & Embedding:

   The image is split into patches (Sentences). Those patches are immediately split into sub-patches (Words). These are converted into mathematical vectors called "embeddings."
2. The Inner Loop (T_{in}):

   The Inner Transformer looks at the "Visual Words" independently within each sentence.

- *Action:* It calculates how specific details relate to each other. For example, in a patch containing a face, it figures out how the "eye" sub-patch relates to the "nose" sub-patch.

3. Aggregation & Augmentation:

   This is the crucial mixing step. The detailed features found by the Inner Transformer are projected (resized) and added to the main Sentence Embedding.
   - *Result:* The "Sentence" representation is no longer just a coarse blob; it is now enriched with fine-grained detail.
4. The Outer Loop (T_{out}):

   The Outer Transformer takes these enriched Sentence Embeddings and models the relationships across the whole image (e.g., how the person on the left relates to the background on the right).

Results

- **Preservation of Local Structure:** Unlike standard Visual Transformers (ViT) or DeiT, which often "corrupt" or ignore the pixel arrangement inside a patch, TNT preserves it by processing the sub-patches (words) explicitly.
- **Efficiency:** Even though it uses two transformers, the computational cost is negligible. The weights for the Inner Transformer are shared across all sentences, meaning the model gets smarter without becoming significantly heavier or slower.
- **Improved Performance:** By extracting features at a fine granularity, the model provides a richer representation of the image, leading to better recognition accuracy.

---

3. Complexity Analysis & Performance Benchmarks

TNT achieves significantly higher accuracy than standard visual transformers (like DeiT) across classification, detection, and segmentation tasks while incurring only a negligible increase in computational cost (approx. 1.14x FLOPs).

Key Concepts & Definitions

- **FLOPs (Floating Point Operations):** A measurement of the computational "work" required by the model. Lower FLOPs mean the model is lighter and theoretically faster.
- **Throughput:** The speed at which the model processes data, measured in images per second (im/s).

- **Downstream Tasks:** Applying the model to specific, practical jobs (like detecting objects in a video) after it has been trained on general images.
- **The Efficiency Trade-off:** The balance between how smart a model is (Accuracy) and how heavy it is (Complexity). TNT improves the "smartness" without adding much "weight."

Complexity Calculation

The efficiency of TNT comes from the mathematical design of its blocks. Even though it adds a second transformer (T_{in}), it remains lightweight because the "word" dimensions are kept small.

1. The Complexity Formula

The total computational cost ($FLOPs_{TNT}$) is the sum of the Inner Transformer, Outer Transformer, and the connection layer:

$$FLOPs_{TNT} = 2nmc(6c + m) + nmcd + 2nd(6d + n)$$

Where:

- n: Number of visual sentences (patches).
- m: Number of visual words (sub-patches).
- c: Dimension of the *word* embedding (small).
- d: Dimension of the *sentence* embedding (large).

2. Why it is efficient

The formula shows that the increase in complexity is low because c (word dimension) is significantly smaller than d (sentence dimension).

- **Standard Block vs. TNT Block:**
  - Standard Block FLOPs: ~376M
  - TNT Block FLOPs: ~429M
  - **Ratio:** TNT is only **1.14x** heavier than a standard block, with only **1.08x** more parameters.

Results & Conclusion

TNT proves that modeling local details leads to better performance without sacrificing speed.

1. ImageNet Classification (The Benchmark)

TNT consistently beats other transformers of similar size.

Model Variant	Params (M)	FLOPs (B)	Top-1 Accuracy
DeiT-S	22.1	4.6	79.8%
TNT-S	**23.8**	**5.2**	**81.5%**
T2T-ViT	63.9	13.2	82.2%
TNT-B	65.6	14.1	82.9%

- **Key Finding:** TNT-S improves accuracy by **1.7%** over DeiT-S with very similar computational costs.

2. Speed vs. Flexibility

- **High Throughput:** TNT-S processes 428 images/second.
- **Tunable Speed:** The model is flexible. For example, **TNT-S-4** (using fewer TNT blocks) matches DeiT-S speed (822 im/s) but still achieves higher accuracy (80.8%).

3. Superior Generalization

Because TNT understands fine details (like edges and textures via the Inner Transformer), it performs exceptionally well when transferred to specific tasks:

- **Object Detection (COCO):** TNT-S outperforms PVT-Small by **3.5 AP** (Average Precision).
- **Semantic Segmentation (ADE20K):** TNT-S scores **43.6% mIoU**, beating both PVT and DeiT-S.
- **Complex Frameworks:** When used in Faster RCNN, TNT-S significantly surpasses the industry standard ResNet-50.
