African Institute for Mathematical Sciences — Cameroon
Structured Master's, Data Science Stream · Computer Vision

Attention, Words and Images

Week 2 · Day 5: Vision Transformers, Text-Guided Localisation and Image Captioning

Yusuf Brima, Ph.D.

Introduction to Computer Vision · AIMS Cameroon, 2026–2027

Today

1. **Attention**: letting every part of an image look at every other part
2. **The Vision Transformer (ViT)**: an image as a sentence of patches
3. **What transformers attend to**, and learning without labels
4. **Localisation from language**: finding "a yellow taxi"
5. **Image captioning**: from pixels to sentences
6. **Week 2 wrap-up and the capstone projects**

By the end of today you should be able to

- compute self-attention for a small example by hand;
- describe the ViT pipeline from patches to a class;
- compare CNNs and ViTs and say when each is preferable;
- explain how text-guided detection and captioning connect vision with language;
- plan your capstone project.

Where we are

This week

- Days 1–2: CNNs, residual networks, transfer learning.
- Days 3–4: detection and segmentation, all built on convolutional backbones.

Today

- A different building block, **attention**, that now powers the largest vision models: SAM's encoder (yesterday), CLIP (Week 1), and multimodal assistants.
- And the first bridge between images and **language**, the theme of Week 3.

Intuition

A convolution asks: "what do my *neighbours* look like?". Attention asks: "which parts of the *whole image* are relevant to me, right now?".

Part 1

Attention

What convolution cannot do easily

- A 3×3 kernel only mixes neighbouring pixels. To relate a person's head to their feet, information must travel through many layers (Day 2: receptive fields grow slowly).
- The kernel's weights are **the same for every image**: it cannot decide, for *this* image, which far-away region matters.

Intuition

To recognise the occluded cat (Week 1, Day 1), it helps if the visible ear can directly "consult" the visible tail across the rock. In a CNN they meet only in the deepest layers.

Practitioner's Note

The bilateral filter (Week 1, Day 3) already had weights that depend on the image: "average with neighbours that *look like me*". Attention generalises that idea to the whole image, with learned similarity.

Transformers came from language

- *Attention Is All You Need* (Vaswani et al., 2017): a model for translating sentences, built only from attention and small MLPs.
- In "The farmer sold the beans because *they* were ripe", the word "they" must attend to "beans", several words away.
- Transformers then took over language: translation, search, and today's large language models.

Intuition

A sentence is a sequence of tokens; so is an image cut into patches. ViT (2020) asked: what if we use the language architecture almost unchanged, and only change how the tokens are made?

Practitioner's Note

Because vision and language now share one architecture, joining them (captioning, CLIP, multimodal assistants) became far easier. That is the road to Week 3.

Self-attention: queries, keys, values

Each token (here: an image patch) produces three vectors with learned matrices:

- a **query** q: "what am I looking for?"
- a **key** k: "what do I contain?"
- a **value** v: "what will I pass on if chosen?"

Each token compares its query with every key, turns the scores into weights with softmax, and takes the weighted average of the values.

Scaled dot-product attention

$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}\Big(\frac{QK^\top}{\sqrt{d}}\Big)\, V$$

d: vector length. Dividing by $\sqrt{d}$ keeps the scores from growing too large.

Intuition

A library: your question (query) is compared with every book's catalogue entry (key); you read the matching books (values), more from the better matches.

Worked example: attention with three patches

Three tokens, 2-D vectors, $Q = K = V = X$

Two patches of cat fur, one of leaf:

$$X = \begin{bmatrix} 1 & 0 \\ 0.9 & 0.2 \\ 0 & 1 \end{bmatrix} \begin{matrix} \text{fur} \\ \text{fur} \\ \text{leaf} \end{matrix}$$

Scores $XX^\top/\sqrt{2}$, row for patch 1: $(0.71,\ 0.64,\ 0)$.

Softmax: $(0.41,\ 0.38,\ 0.20)$.

Output: $0.41(1,0) + 0.38(0.9, 0.2) + 0.20(0,1) \approx (0.76,\ 0.28)$.

- Patch 1 takes most of its new value from itself and the *other fur patch*, wherever it is in the image.
- The leaf patch's row is $(0.24,\ 0.28,\ 0.49)$: it listens mostly to itself.
- Similar patches pool their information; different ones stay apart. A learned, image-specific grouping (Gestalt similarity, Week 1, Day 1).

Two details, worked: shapes and the $\sqrt{d}$

Shapes in one attention head of ViT-B

196 patch tokens (+1 [CLS]), head dimension $d = 64$:

Q, K, V: 197×64 each.

$QK^\top$: 197×197 scores.

softmax per row, then $\times V$: 197×64.

12 heads side by side: 197×768, mixed by one more 768×768 matrix.

Why divide by $\sqrt{d}$?

If q and k have d independent entries of variance 1, their dot product has variance d.

$d = 64$: scores spread with standard deviation 8, so softmax puts almost all weight on one token and gradients vanish.

Dividing by $\sqrt{64} = 8$ brings the spread back to about 1.

Intuition

Without scaling, attention becomes a hard "pick one" instead of a soft "mix several", and learning stalls: the same saturation problem as sigmoids on Day 1.

Multi-head attention and its cost

- One attention pattern captures one kind of relation. **Multi-head** attention runs several (e.g. 12) in parallel with different learned Q, K, V matrices, then combines them.
- One head may link parts of the same object, another may follow colour, another position.

Attention is quadratic

Every token attends to every token: n^2 scores per head.

224×224 image, 16×16 patches: $n = 196$; $196^2 \approx 38\,000$ scores.

1024×1024 image: $n = 4096$; $4096^2 \approx 16.8$ million per head per layer.

Practitioner's Note

This is why ViTs use large patches, and why high-resolution models (Swin, SAM) restrict attention to local windows.

Attention and convolution, side by side

	Convolution	**Self-attention**
Who interacts	neighbours in a $K \times K$ window	every token with every token
Weights	fixed after training	computed for each image (from Q, K)
Position	built in (sliding window)	must be added (position embeddings)
Cost per layer	linear in the number of pixels	quadratic in the number of tokens
Built-in assumptions	locality, shift-equivariance	almost none

Intuition

Convolution is a fixed, local template; attention is a flexible, global, image-specific weighting. The bilateral filter (Week 1, Day 3) and squeeze-and-excitation (Day 2) were steps between the two.

Part 2

The Vision Transformer

ViT (2020): an image is worth 16×16 words



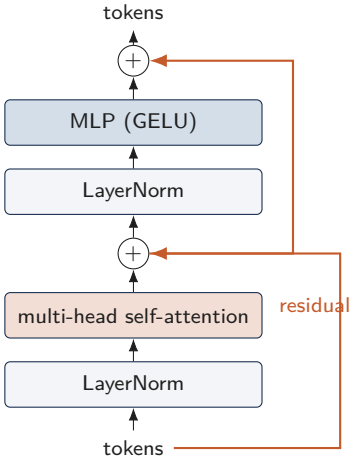
RyanDannyChacko, CC BY-SA 4.0

1. Cut the image into 16×16 patches: $14 \times 14 = 196$.
2. Flatten each patch ($16 \cdot 16 \cdot 3 = 768$ numbers) and project it linearly to a token vector.
3. Add a learned **position embedding** to each token.
4. Prepend a special **[CLS]** token.
5. Run a standard transformer encoder; classify from the final [CLS].

Intuition

The patch projection is exactly a convolution with a 16×16 kernel and stride 16. After that, there is no convolution at all: every patch can attend to every other from the very first layer.

The transformer encoder block



- **Attention**: tokens exchange information across the image.
- **MLP**: each token is processed on its own (like a 1×1 conv).
- **Residual connections** (Day 2) and **normalisation** (Day 1) make 12–32 blocks trainable.

ViT-Base

12 blocks, token width 768, 12 heads of 64 dimensions, MLP width 3072: about 86.6 million parameters.

Where ViT-B's 86 million parameters are

One block, width 768

Attention: Q, K, V and output projections, $4 \times 768 \times 768 \approx 2.36$M.
MLP: $768 \times 3072 + 3072 \times 768 \approx 4.72$M.
Per block: about 7.1M.

12 blocks: about 85M.
Patch projection (768×768) + position embeddings (197×768): about 0.7M.
Total: about **86 million**.

- Two thirds of the parameters are in the MLPs, not in attention.
- The number of parameters does not depend on the image size; the computation does (quadratically).

Intuition

Attention *routes* information between tokens; the MLP *processes* it inside each token. Most of a transformer's knowledge is stored in its MLPs.

Patch size: detail vs. cost

A 224×224 image

Patch	Tokens	Scores per head
32×32	49	2 401
16×16	196	38 416
8×8	784	614 656

- Halving the patch size gives 4 times more tokens and 16 times more attention computation.
- Small patches see finer detail (better for segmentation and small objects); large patches are cheap (fine for classification).

Intuition

A 16×16 patch is the ViT's "pixel": detail inside one patch is only seen through the linear projection, never through attention.

Design details: [CLS], LayerNorm, resolution

[CLS] token or average?

- The [CLS] token attends to all patches and becomes the image summary.
- Averaging all patch tokens (like global average pooling) works about as well and is used by many recent models.

LayerNorm, not BatchNorm

- LayerNorm normalises each token over its own 768 features, independent of the batch.
- Works for any batch size, including one image at a time.

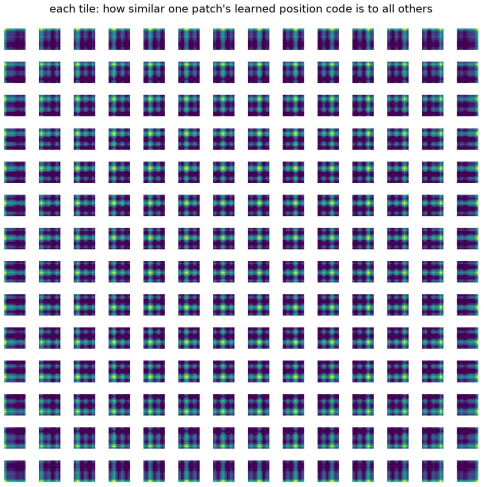
Changing the image size

- Train at 224×224 (196 tokens), fine-tune at 384×384 (576 tokens): the learned position embeddings are **interpolated** from a 14×14 to a 24×24 grid.
- That is how we fed 480×480 images to DINO for the attention maps.

Practitioner's Note

Higher resolution at fine-tuning often adds a point or two of accuracy, at several times the cost.

Position embeddings: where is each patch?



- Attention by itself ignores order: shuffle the patches and the output is shuffled in the same way. The model would not know the sky is above the road.
- So each patch position gets a **learned** vector added to its token.

Real weights of ViT-B/16. Each small tile shows, for one patch position, how similar its position code is to every other position's. Bright spots sit around the patch itself, and along its row and column.

Intuition

Nobody told the model about rows and columns. It discovered the 2D grid from data.

CNNs and ViTs: built-in assumptions vs. data

CNN

- Built-in: locality and shift-equivariance (Week 1, Day 3).
- Learns well from thousands of images.
- Receptive field grows slowly with depth.

ViT

- Few built-in assumptions: must *learn* locality and position.
- Needed hundreds of millions of images (JFT-300M) to beat ResNets in 2020; DeiT (2021) reduced this with strong augmentation and distillation.
- Global context from layer 1; scales very well with data.

Model	Params	Top-1	GFLOPs
ResNet-50	25.6M	76.1	4.1
ConvNeXt-T	28.6M	82.5	4.5
ViT-B/16	86.6M	81.1	17.6
Swin-T	28.3M	81.5	4.5

ImageNet-1k, torchvision model cards.

Intuition

Assumptions are a substitute for data. With little data, built-in assumptions help; with huge data, the model can learn better ones.

Training ViTs without 300 million images

- The original ViT needed Google's private JFT-300M dataset to beat ResNets.
- **DeiT** (2021) trained ViTs on ImageNet alone, on one machine, with:
  - very strong augmentation (random erasing, mixing images and labels);
  - longer training and careful regularisation;
  - *distillation*: learning also from a CNN teacher's predictions.
- DeiT-B: about 82% top-1 on ImageNet with no extra data.

Intuition

A CNN's built-in assumptions can be *taught* instead of built in: augmentation shows the model which changes do not matter; a CNN teacher passes on its locality bias through its predictions.

Practitioner's Note

For your own data, you will almost never train a ViT from scratch. Fine-tune a pre-trained one, exactly like Day 2's ResNet.

Swin and hybrid designs

- **Swin Transformer** (2021): attention only inside small windows, which shift between layers; merge patches to build a pyramid, like a CNN.
- Cost grows linearly with image size, so it works for detection and segmentation backbones.
- **ConvNeXt** (Day 2) went the other way: a CNN redesigned with transformer ideas.

Intuition

CNNs and transformers have converged: the best modern designs mix local operations (cheap, good for detail) with some global attention (context). The labels matter less than the principles from this week.

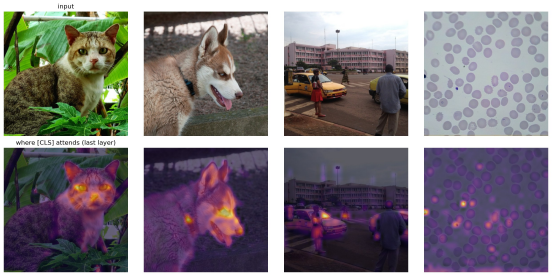
Practitioner's Note

For a capstone: a pre-trained ConvNeXt, Swin or ViT from `timm` or torchvision, fine-tuned like Day 2's ResNet. The code is the same.

Part 3

What do transformers attend to?

Attention maps from a self-supervised ViT (DINO)



Photos: RyanDannyChacko; MrPanyGoff; Bdx; OpenFlexure Project (CC BY-SA)

DINO ViT-S/16 (2021), trained **without any labels**. Bottom row: where the [CLS] token attends in the last layer, averaged over heads. Cat, husky, people and taxis, and the stained parasites stand out from the background.

Intuition

Nobody gave DINO masks, boxes or class names, yet its attention separates objects from background. Self-supervised learning (Week 1, Day 1) discovers "objectness" from images alone.

Different heads, different questions



Photokadaffi, CC BY-SA 4.0

The six attention heads of DINO's last layer, each shown separately on the Mbouda street scene. Each head attends to a different set of regions: some to people, some to the cargo or the motorbike, some to the background.

Intuition

Multi-head attention in action: several "questions" asked in parallel, each grouping the image differently. Their combination gives the model a rich, overlapping description of the scene.

Fine-tuning a vision transformer in code

```
import timm, torch
model = timm.create_model("vit_base_patch16_224",
                          pretrained=True,
    num_classes=3)
cfg = timm.data.resolve_data_config({}, model=model
    )
transform = timm.data.create_transform(**cfg)  #
    224, ViT mean/std

# freeze everything except the last 2 blocks and the
     head
for name, p in model.named_parameters():
    p.requires_grad = name.startswith(("blocks.10",
     "blocks.11",
                                       "norm", "
    head"))
opt = torch.optim.AdamW(
    [p for p in model.parameters() if p.
    requires_grad],
    lr=1e-4, weight_decay=0.05)
# ... the same training loop as Day 1
```

- Exactly Day 2's recipe: new head, freeze most layers, small learning rate.
- ViTs usually prefer AdamW (Adam with decoupled weight decay) and a warm-up of the learning rate.
- Use the model's own normalisation statistics: they differ from ImageNet's ResNet values.

Learning without labels, revisited

Masked autoencoders (MAE, 2021)

- Hide 75% of the patches at random; the encoder sees only the remaining 49 of 196.
- A light decoder must reconstruct the missing patches.
- To fill in a hidden cat's head from its ears and tail, the model must learn what cats look like: amodal completion (Week 1, Day 1) as a training task.

DINO (2021): self-distillation

- Two networks: a *student* and a *teacher* (a slowly updated average of the student).
- Each sees a different crop of the same image; the student must match the teacher's output.
- "These two crops are the same scene" is the only supervision. Objectness emerges, as we saw.

Intuition

Labels are expensive; images are nearly free. Self-supervised pre-training on millions of unlabelled images, then fine-tuning on a few hundred labelled ones, is the main recipe for low-label settings, such as many African agricultural and medical applications.

Foundation models built on vision transformers

Model	Year	Trained on	Used in this course
CLIP	2021	400M image–caption pairs	zero-shot classification (Week 1, Day 1)
DINO / DINOv2	2021/23	unlabelled images	attention maps today; general features
SAM	2023	1.1B masks	one-click masks (Day 4)
Depth Anything	2024	labelled + pseudo-labelled depth	depth from one photo (Week 1, Day 1)
OWL-ViT	2022	CLIP + detection data	text-guided boxes (today)

Intuition

One architecture, pre-trained once at enormous scale, reused for many tasks. Week 3 asks what this means for cost, access, and who gets to build such models.

Are ViTs more robust to occlusion?

The occluded cats from Week 1, Day 1, again

Photo	ResNet-50	ViT-B/16
behind leaves	Egyptian cat	Egyptian cat
behind a rock	weasel	badger
behind lace	vestment	overskirt
camouflaged leopard	cheetah	cheetah

- Both models were trained on ImageNet; the ViT (from Hugging Face, also pre-trained on ImageNet-21k) has seen more data.
- On these four photos the ViT makes the same kinds of mistakes as the CNN: wrong on the rock, the lace and the camouflaged leopard.
- Larger studies find ViTs somewhat more robust to occlusion and more shape-biased than CNNs, but neither is human-like.

Note: four photos are an illustration, not a study; compare models on hundreds of images per condition.

Part 4

Localisation from language

Localisation: from a fixed class list to any description

- **Classification + localisation** (Day 3): one object, a class and a box.
- **Detection** (Day 3): many objects, but only the classes seen in training (80 for COCO).
- **Open-vocabulary detection**: describe what you want in words, "a yellow taxi", "a woman in a red dress", and get boxes, even for phrases never in a training list.

Intuition

CLIP (Week 1, Day 1) learned to put an image and its caption close together in one space. Do that for every *patch* instead of the whole image, and each patch can be matched to a phrase: that is a detector driven by text.

OWL-ViT (2022), in one line

A ViT gives a vector and a box for every patch; a text encoder gives a vector for each query phrase; score = similarity between the two.

Classification plus localisation, worked

- One object per image (a leaf, an X-ray, a document): add a box head with 4 outputs to a classifier (Day 3, step 1).
- Coordinates normalised to $[0, 1]$, trained with an L_1 or smooth-L_1 loss, plus cross-entropy for the class.

One training example

True box $(0.25, 0.40, 0.65, 0.85)$, predicted $(0.30, 0.40, 0.60, 0.80)$.
$L_1 = 0.05 + 0 + 0.05 + 0.05 = 0.15$.
Class loss: $-\log p_{\text{correct}}$, e.g. $-\log 0.8 = 0.22$.
Total with $\lambda = 1$: 0.37.

Intuition

Useful whenever there is exactly one thing to find: the leaf in a phone photo, before cropping and classifying it; the lungs in an X-ray; the document page in a scan.

Practitioner's Note

Cropping to the localised region, then classifying the crop at higher resolution, often beats classifying the whole image: a simple two-step pipeline.

How text-guided detection works

1. Start from CLIP: an image encoder (ViT) and a text encoder, trained to match images and captions.
2. Keep the ViT's output token for *every patch*, not only [CLS]. Add a small head that predicts a box for each token.
3. Compare each token's embedding with the embedding of each query phrase.
4. Fine-tune on detection datasets so that tokens' boxes and similarities become accurate.

Intuition

Every patch proposes "an object around here, and it looks like *this* in the shared image–text space". The query phrase picks the matching proposals. Language replaces the fixed list of 80 classes.

Practitioner's Note

Phrase queries naturally: "a photo of a . . . " and plain nouns work better than long sentences. Check scores against a few hand-labelled images before trusting a threshold.

Text-guided localisation on real photographs



Photos: Bdx; Photokadaffi (CC BY-SA 4.0)

OWL-ViT (base), queries typed at test time. It found “a yellow taxi”, “a lamp post”, “a bunch of bananas” and “a large white sack”, none of them COCO classes. It found *no* box for “a woman in a red dress” above our 0.1 threshold: attribute phrases are harder than nouns. Scores are similarities, not probabilities.

Practitioner’s Note

Open-vocabulary detectors are excellent for exploring a new problem and for pre-labelling data. For a deployed system with fixed classes, a fine-tuned YOLO (Day 3) is usually faster and more accurate.

Zero-shot or fine-tuned?

Zero-shot (open-vocabulary) models

- No labels needed; any phrase.
- Good for exploring, prototyping and pre-labelling.
- Weaker on unusual domains (microscopy, X-rays, satellite) and on fine distinctions (two crop diseases).

Fine-tuned models

- Need labels for your classes.
- Faster, smaller, more accurate on those classes.
- Know nothing outside them.

Intuition

The best of both: use a zero-shot model to propose labels, have an expert check them quickly, then fine-tune a small model on the checked labels (the data engine of Day 4).

Referring expressions

"The woman in the red dress *next to* the taxi" needs relations, not just nouns. Grounding models (e.g. Grounding DINO) handle such phrases better, but test them: relations are where they fail most.

Part 5

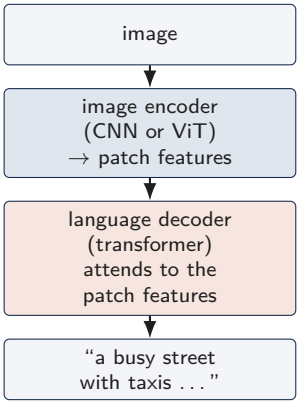
Image captioning

The captioning task

- **Input**: an image. **Output**: a sentence describing it.
- Much harder to evaluate than a class: many sentences are correct.
- Needs *what* (objects), *where* (relations: "on", "next to") and *language* (grammar, word choice).

Uses

- Describing scenes to blind and low-vision users.
- Searching photo archives by content.
- Drafting reports from images (with expert review).



Where captioning models learn

- **COCO Captions** (2015): about 123 000 images, each with 5 captions written by crowd workers.
- **Flickr30k**: about 31 000 photos, 5 captions each.
- **Web image–text pairs**: hundreds of millions of images with their alt-text, noisy but huge (BLIP filters and re-captions them).

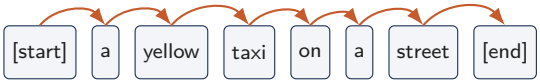
Intuition

Five people describe the same photo in five different ways: captioning has no single right answer. The model learns what people *typically* mention: the main objects and actions, rarely the details an expert would.

Practitioner's Note

Crowd workers and web users wrote these captions, mostly in English and mostly about everyday Western scenes. Expect the model's vocabulary and assumptions to reflect that.

How a caption is generated, word by word



1. The decoder reads the image features and the words so far.
2. It outputs a probability for every word in its vocabulary (softmax over, say, 30 000 words: Day 1's classifier).
3. Pick a word, append it, repeat until [end].

Training: cross-entropy on the next word of human-written captions.

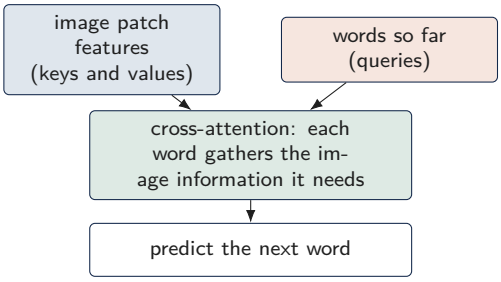
Intuition

Captioning is classification, repeated: "which word comes next?". While choosing each word, the decoder attends to the image patches most relevant to it: "taxi" looks at the taxi, "street" at the road (Show, Attend and Tell, 2015).

Practitioner's Note

Beam search keeps the best few partial sentences instead of only the single best word at each step.

Cross-attention: words looking at patches



- **Self-attention**: queries, keys and values all come from the same tokens (today's earlier slides).
- **Cross-attention**: queries come from the *text*, keys and values from the *image*.
- The same formula, $\mathrm{softmax}(QK^\top/\sqrt{d})V$; only the source of Q changes.

Intuition

When the decoder is about to write "taxi", its query matches the keys of the taxi patches, and their values flow into the prediction.

Greedy decoding vs. beam search, worked

Two words, two candidates each

First word: "a" 0.5, "the" 0.4.
After "a": "taxi" 0.3, "car" 0.3, others 0.4.
After "the": "taxi" 0.9.

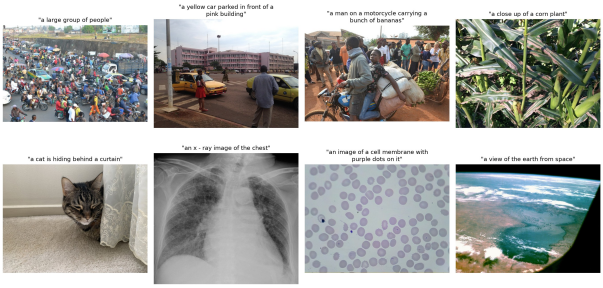
Greedy: picks "a" (0.5), then "taxi": probability $0.5 \times 0.3 = 0.15$.
Beam search (keep 2): "the taxi" has $0.4 \times 0.9 = 0.36$: better overall.

- Greedy: always take the most likely next word. Fast, but can walk into a dead end.
- Beam search: keep the B best partial sentences at each step (we used $B = 3$ for BLIP).
- Sampling: pick words at random by probability. More varied, less reliable.

Intuition

The best sentence is not always built from the best word at each step, much as the best route home does not always start with the shortest first street.

BLIP describes this course's photographs



Photos (Wikimedia Commons): Minette Lontsie; Bdx; Photokadaffi; Chroanch; Famartin; B. Herreros et al.; OpenFlexure Project; NASA

BLIP-base (2022), captions generated on a laptop CPU, unedited. Read them critically: which are right, which are vague, which are wrong?

Intuition

Street scenes, crowds and the cat: fluent and mostly right. The X-ray, the blood smear and the view of Lake Chad from space test whether the model knows these domains.

Reading the captions critically

- **Douala crossroads**: "a large group of people"
- **Yaoundé street**: "a yellow car parked in front of a pink building"
- **Mbouda moto-taxi**: "a man on a motorcycle carrying a bunch of bananas"
- **Maize blight**: "a close up of a corn plant"
- **Cat and curtain**: "a cat is hiding behind a curtain"
- **Chest X-ray**: "an x - ray image of the chest"
- **Blood smear**: "an image of a cell membrane with purple dots on it"
- **Lake Chad from Apollo 7**: "a view of the earth from space"

Practitioner's Note

Captions inherit the training data: web photos with alt-text. Medical images, microscopy and satellite views are rare there, so expect generic or wrong answers. Never use a general-purpose captioner for clinical reports.

Intuition

Notice what captions leave out: the disease on the leaf, the parasites in the smear. A caption describes what is *typical to say*, not what an expert needs.

Visual question answering

- **Input**: an image and a question. **Output**: an answer.
- "How many people are on the motorbike?" "What colour is the taxi?" "Is this leaf diseased?"
- Same machinery as captioning: the question tokens attend to the image, and the decoder generates the answer.
- VQA v2 (2017) carefully paired images so that the answer to the same question differs, to stop models from answering from the question alone.

Intuition

Early VQA models answered "yes" to "Is there a . . . ?" most of the time and scored well, because most such questions in the data had the answer "yes". A shortcut again (Week 1, Day 5).

Practitioner's Note

Counting questions remain hard for many vision–language models. Compare with Day 3: a detector plus counting is often more reliable than asking "how many?".

Bias and hallucination in captions

Bias

- Captioners have been shown to guess a person's gender from context (a kitchen, a snowboard) rather than from the person (Hendricks et al., 2018).
- Models can *amplify* correlations in their training data (Zhao et al., 2017).
- Scenes from under-represented regions may be described vaguely or with stereotypes.

Hallucination

- Captions sometimes mention objects that are not in the image but usually co-occur with what is (a "fork" next to any plate).
- The text is fluent and confident either way.

Practitioner's Note

Before deploying a captioner, read 100 of its captions on your own images and count errors by type: wrong object, invented object, wrong attribute, biased wording.

A use that matters: describing images to people who cannot see them

- Hundreds of millions of people live with visual impairment worldwide, a large share of them in low- and middle-income countries.
- Phone apps that read text aloud, describe scenes and identify banknotes or medicines already use captioning, OCR and detection.
- Local needs: local languages, local currency, local products and signage.

Intuition

Here a caption is not a benchmark score but someone's view of their surroundings. "A busy street with taxis" is less useful than "a taxi is stopping in front of you on the right".

Practitioner's Note

A strong capstone direction: pick one narrow, real need (reading a medicine label, identifying a banknote), collect local images, and evaluate with the people who would use it.

Evaluating captions, and what comes next

Automatic metrics

- **BLEU**, **CIDEr**: word overlap with several human captions. Cheap, but a correct caption in different words scores low.
- **CLIPScore**: image–caption similarity from CLIP.

Human evaluation

- Is it correct? Complete? Useful for the person who will read it?

From captioning to multimodal assistants

- Connect a vision encoder to a large language model: ask questions about an image, in any language, and get answers (2023 onwards).
- Same principle: image patches become tokens the language model attends to.
- Week 3 covers CLIP and vision–language models in depth, and their risks.

Practitioner's Note

Fluent text is not correct text. Language models describe images confidently even when wrong ("hallucination"). Always test on images from your own domain.

Part 6

Week 2 wrap-up and the capstone

CNN or ViT for your project?

Situation	Start with	Why
Few hundred labelled images	pre-trained ResNet/ConvNeXt or ViT, frozen or lightly fine-tuned	both work; compare quickly
CPU or phone deployment	MobileNet, EfficientNet, small YOLO	efficient, mature tooling
Need global context, plenty of data	ViT or Swin	attention scales with data
Very few labels, many unlabelled images	DINOv2 features + linear classifier	self-supervised features transfer well
Language queries or descriptions	CLIP, OWL-ViT, BLIP	shared image–text space

Practitioner's Note

Do not choose by fashion. Try two candidates on your validation set with the same data and training budget, and keep the simpler one if they tie.

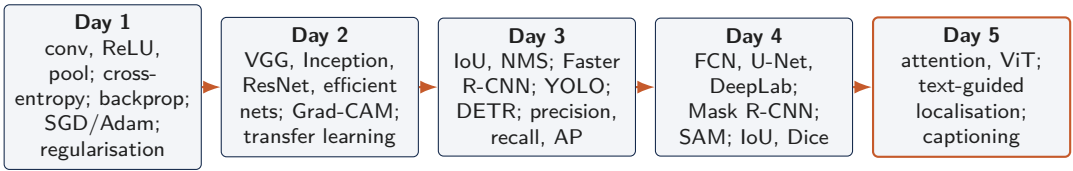
Week 1 ideas inside Week 2 models

Week 1 idea	Where it reappeared this week
convolution, kernels as templates	every CNN layer; ViT's patch projection
normalisation (Day 2)	input statistics; batch norm; layer norm
blur before downsampling (aliasing)	strided convolutions and pooling; why CNNs are not perfectly shift-invariant
image pyramids, scale space	feature pyramids in detection; U-Net's levels
non-maximum suppression	Canny edges $\rightarrow$ detection boxes
histograms of gradients, pooling	pooling layers; global average pooling
connected components, morphology	post-processing segmentation masks
Gestalt grouping, amodal completion	attention maps; masked autoencoders

Intuition

Deep networks did not replace Week 1; they learned to do Week 1's operations themselves, at scale, and to combine them.

Week 2 in one picture



Intuition

One recipe ran through the week: a pre-trained **backbone** turns pixels into features; a task-specific **head** turns features into labels, boxes, masks or words; a **loss** measures the error; backpropagation does the rest.

Key takeaways for today

1. **Attention**: each token takes a weighted average of all tokens' values, weighted by query–key similarity: $\mathrm{softmax}(QK^\top/\sqrt{d})V$.
2. **ViT**: patches $\rightarrow$ tokens + positions $\rightarrow$ transformer blocks $\rightarrow$ [CLS] $\rightarrow$ class.
3. Fewer built-in assumptions than a CNN: needs more data, scales further.
4. Self-supervised ViTs (DINO) discover objects without labels.
5. Matching patches to text gives **open-vocabulary localisation**.
6. **Captioning** = next-word classification conditioned on the image; read outputs critically, especially outside everyday photos.

The capstone project: what happens now

- Teams of **2–3**; a real-world vision problem; presented on the **last day of Week 3**.
- Proposed topics are released today on the course homepage.
- **Your own idea?** Submit a one-page synopsis by the end of today: problem, motivation, proposed method, and **how you will get the data**.
- Week 3 includes consultation time with me for every team.

The capstone brief (on the course website, under Week 2) gives, for each of 12 topics: why it matters, a real dataset, a minimum deliverable, stretch goals and pitfalls. It also has the timeline, deliverables, rubric and synopsis template.

A good capstone is small, real and honestly evaluated: a clear baseline, a held-out test set from a different source, and an analysis of failures.

Twelve proposed topics

P1	Cassava disease diagnosis from phone photos
P2	Measuring leaf-disease severity (segmentation)
P3	Malaria parasite screening in blood smears
P4	TB screening that generalises across hospitals
P5	Pothole and road-damage detection
P6	Counting vehicles and moto-taxis from street video
P7	Land-cover mapping from Sentinel-2, tested on Cameroon
P8	Flood mapping from satellite images
P9	Wildlife camera-trap triage
P10	CFA franc banknote recognition for blind users
P11	Grading produce ripeness at the market
P12	Helmet-use monitoring with a label-efficient data engine

Intuition

Every topic reuses something from these two weeks: transfer learning (Day 2), detection and AP (Day 3), U-Net and SAM (Day 4), open-vocabulary pre-labelling (today), and Week 1's colour rules as baselines. Several ask you to collect a small *local* test set: that is what shows whether a model works here.

Deliverables and assessment

Deliverables

- **Presentation** on the last day of Week 3: about 10 minutes + 5 minutes of questions; every member speaks; an optional demo video of at most 2 minutes.
- **Code** that reproduces your main numbers, with a README.
- **One-page summary**: question, data, method, results, two failures, limitations, who did what.

Assessment

problem framing	15%
data, labelling, leakage-free splits	20%
baseline and main method	20%
evaluation and metrics	20%
failure analysis, limitations	15%
presentation, reproducibility	10%

Practitioner's Note

An honestly evaluated modest result beats an impressive number from a leaky split. Book a consultation slot early in Week 3.

Capstone timeline

When	What
today	topics released; own-idea synopses due by the end of the day
Week 3, Day 1	teams confirmed; data downloaded and inspected
Week 3, Days 2–3	baseline by Day 2; main model by Day 3; consultations
Week 3, Day 4	evaluation, failure analysis, slides; code frozen
Week 3, Day 5	presentations (a short demo video is welcome)

What to show at the end

- the problem and who it is for;
- the data: source, size, how you split it;
- a simple baseline and your model;
- honest metrics on a held-out test set;
- failure cases and what you would do next.

Intuition

Data first: a team with clean data and a frozen-feature baseline on day two of Week 3 is in a much stronger position than a team still searching for data on day four.

An example one-page synopsis

Detecting bean rust from phone photos

Problem: farmers notice rust late; extension officers are few.

Motivation: beans are a staple; early action saves yield.

Method: frozen ResNet-50 baseline (Day 2), then fine-tuning; Grad-CAM to check the evidence; test on photos from a different region.

Data: iBean (Uganda, open licence) for training; 60 photos collected locally, labelled by an agronomist, for testing.

Success: per-class recall above 85% on the local test photos; a list of failure types.

Intuition

Notice what makes it strong: a real user, a data plan that does not depend on luck, a baseline, a test set from *elsewhere*, and a measurable goal.

Practitioner's Note

Your synopsis does not need a novel method. It needs a clear question and a credible plan to answer it in one week.

Assessment and preparation for Week 3

This week's assessment

- **Quiz 2**: one hour, covers Week 2. Time and room announced by the tutors.
- **Assignment 2**: individual, typed in LaTeX, theory + coding with PyTorch. Due at the start of Week 3, Day 3.

Practise (tutorial with the tutors)

- Compute the attention example in NumPy; then visualise DINO attention on your own photos.
- Caption ten of your photos; classify the errors.

Week 3: generation, applications and the road ahead

- Autoencoders and VAEs
- Diffusion models
- CLIP and vision–language models
- Applications across domains
- Ethics, fairness, responsible deployment
- Capstone work and presentations

Credits

Pre-trained models via Hugging Face: DINO ViT-S/16 (Meta AI), ViT-B/16 (Google), OWL-ViT base (Google), BLIP base (Salesforce). Model-card numbers: torchvision. All outputs were produced for this lecture with `Week2/tools/make_week2_figures.py`.

Photographs from Wikimedia Commons:
`cat_leaves`: RyanDannyChacko, CC BY-SA 4.0
`husky`: MrPanyGoff, CC BY-SA 3.0
`yaounde_taxis`: Bdx, CC BY-SA 4.0
`malaria_smear`: OpenFlexure Project, CC BY-SA 4.0
`moto_mbouda`: Photokadaffi, CC BY-SA 4.0
`douala_ndokoti`: Minette Lontsie, CC BY-SA 4.0
`maize_blight`: Chroanch, CC BY-SA 4.0
`cat_curtain`: Famartin, CC BY-SA 4.0
`xray_tb`: B. Herreros et al., CC BY 4.0
`lake_chad_apollo7`: NASA, Public domain