Solve A-OKVQA with minimal inference-time compute budget

- Given a pre-trained VLM (nanoVLM) with MCQ baseline accuracy of X%
- Reduce the inference-time compute budget as much as possible while still maintaining .85*X% accuracy
  - work on validation set; only final evaluation is on the test set
- You can modify any component of the base VLM
- You can use any data for training
- You can add any components trained from scratch
- You cannot introduce any new pre-trained components

Lab 1

- In class: evaluate lusxvr/nanoVLM on the dev split of A-OKVQA (A) in MCQ setting and (B) in open-answer setting.
  - deliverables:
    - MCQ accuracy
    - OA accuracy
- At home: finetune nanoVLM on the train split of A-OKVQA for the MCQ setting
  - deliverable: MCQ accuracy for 3 hyperparameter configurations

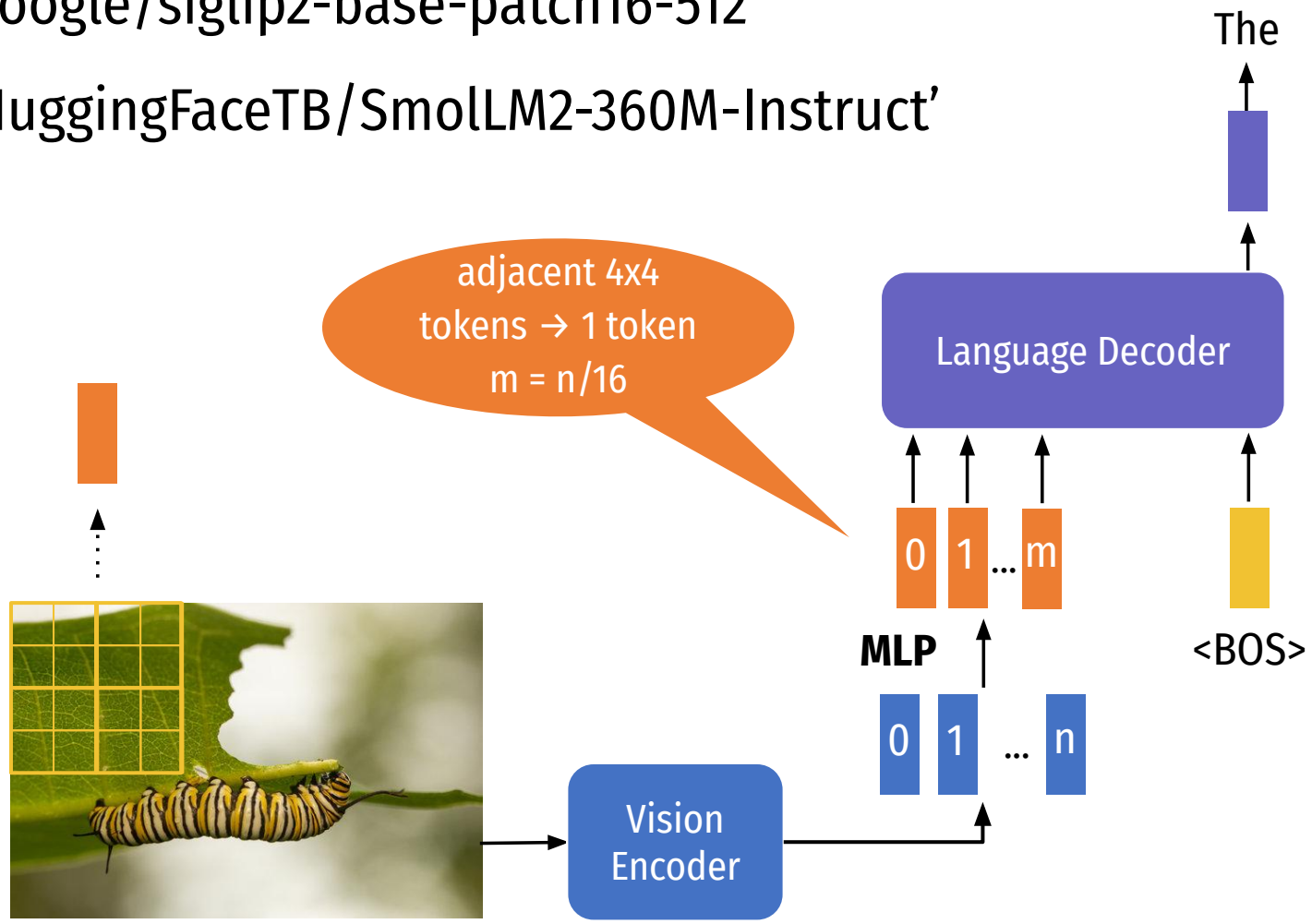
Resources: nanoVLM GitHub repo

Lab 2

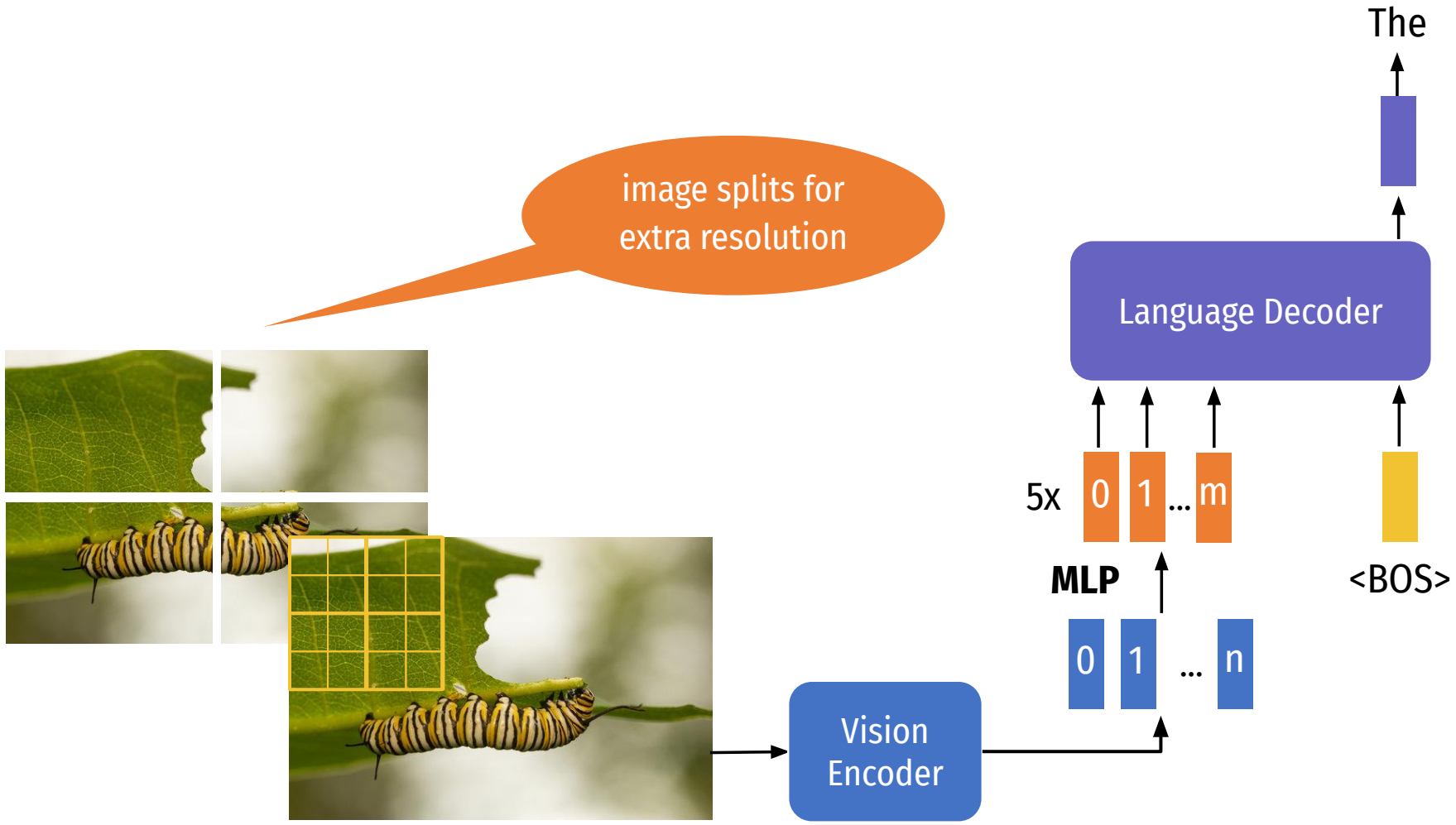
- In class: write code to measure the FLOPs of a forward pass with nanoVLM on: <Image> What is the name of the dog's toy?
  - deliverables:
    - FLOPs for image encoder by hand
    - FLOPs for image encoder automatic
    - FLOPs for 3 generation steps (three tokens) with and without 5 warm up steps
- At home: read the A-OKVQA paper

nanoVLM

- encoder: 'google/siglip2-base-patch16-512'
- decoder: ‘HuggingFaceTB/SmolLM2-360M-Instruct’



nanoVLM


The
Language Decoder
image splits for extra resolution
5x
0
1
...
m
MLP
<BOS>
0
1
...
n
Vision Encoder

DynamicResize

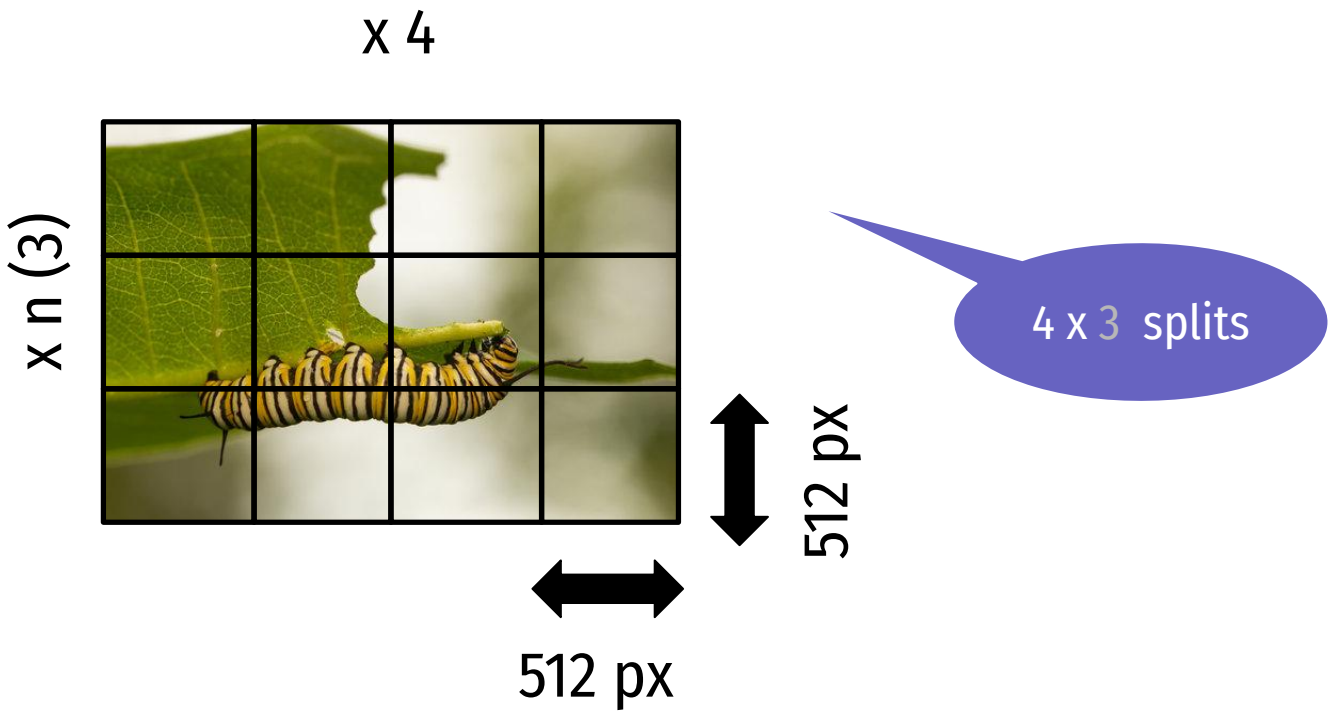
`max_img_size: int = 2048`

`resize_to_max_side_len: bool = True`

2048 px

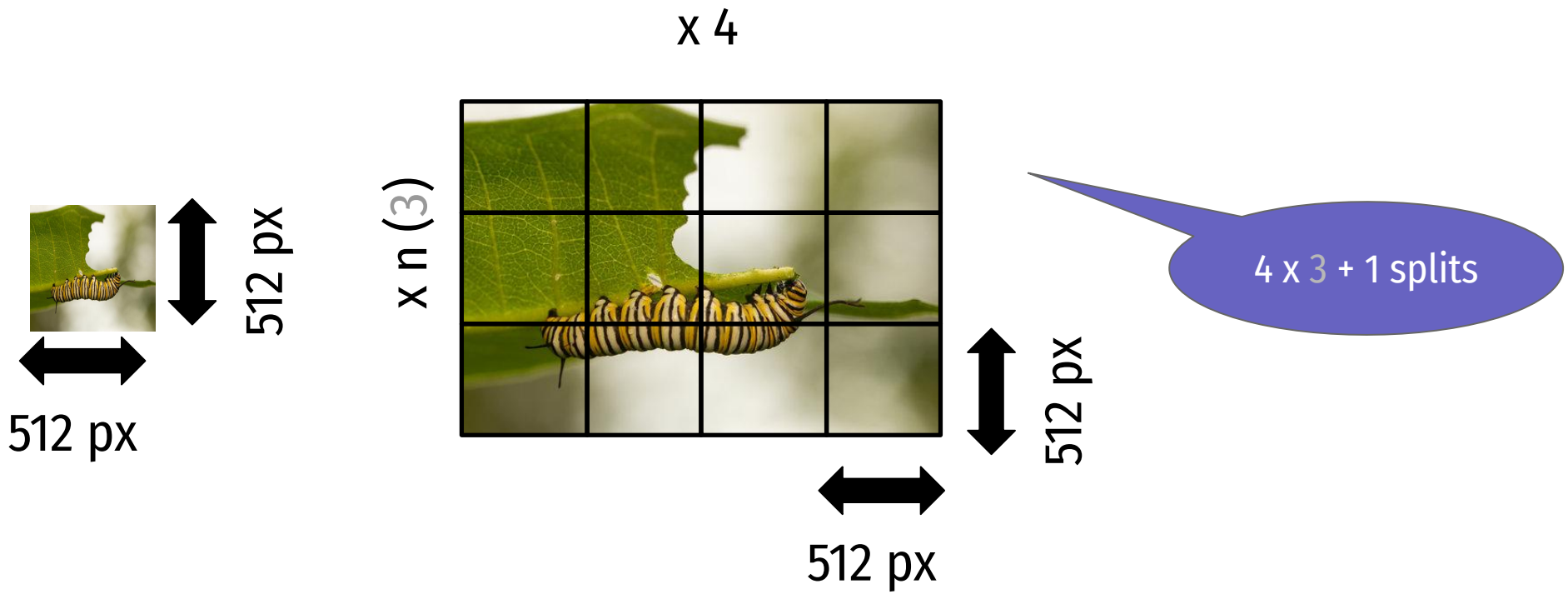
GlobalAnd**SplitImages**

`vit_img_size: int = 512`



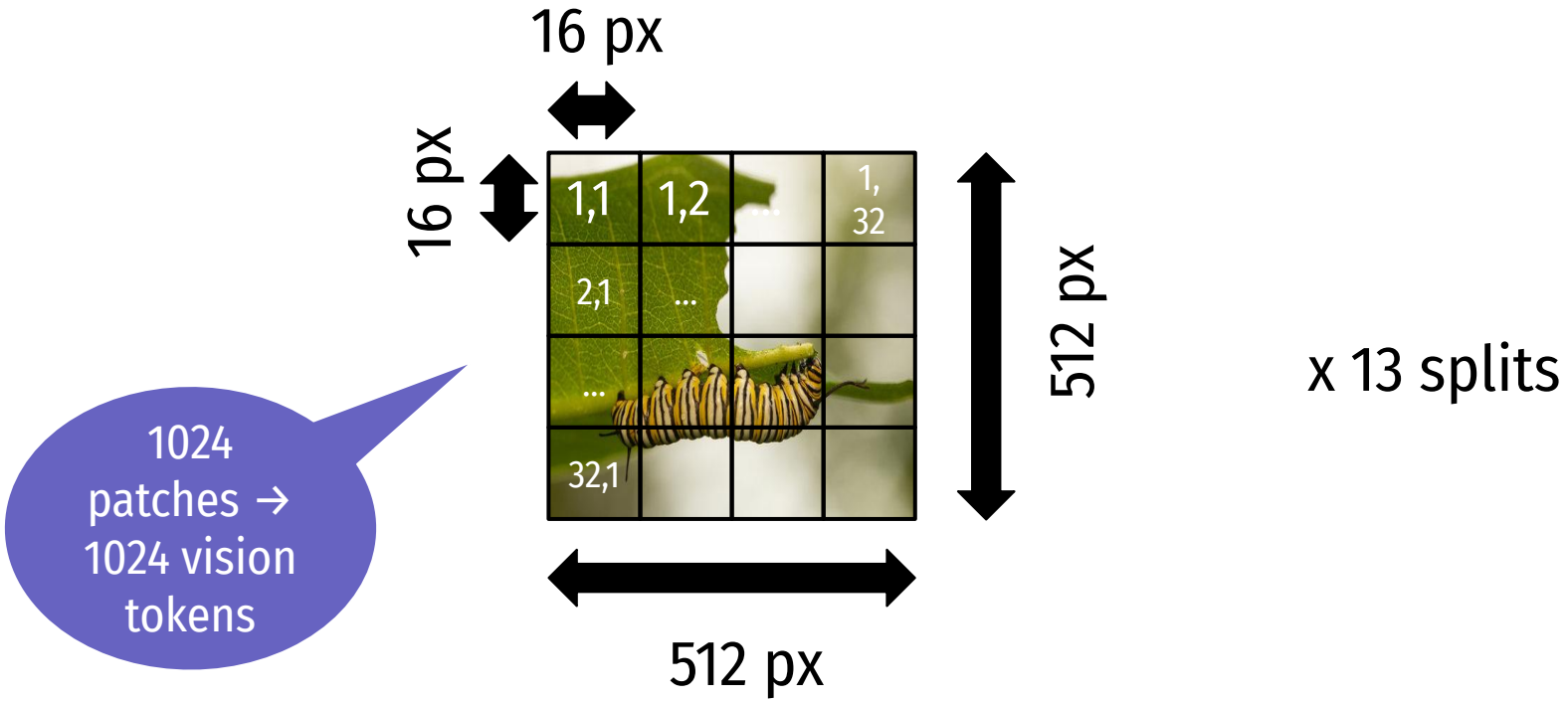
GlobalAndSplitImages

`vit_img_size: int = 512`



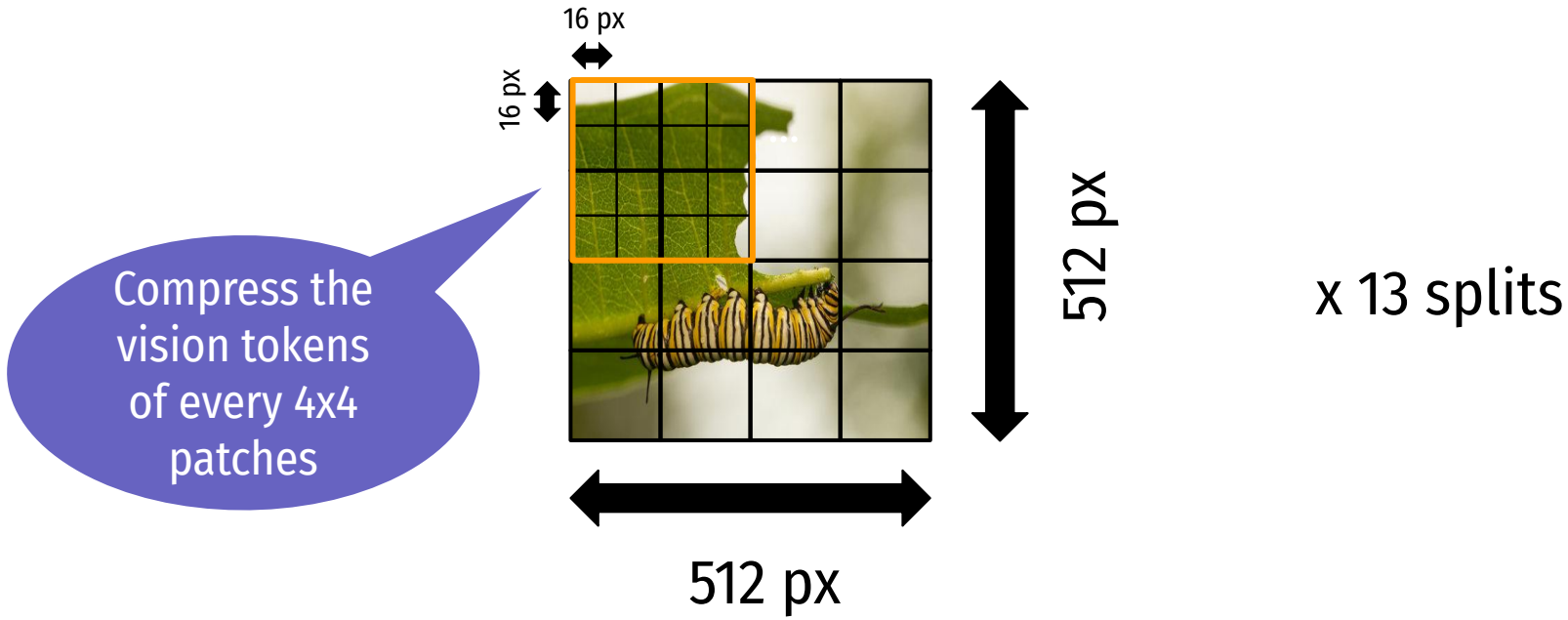
VisionTransformer

```
vit_patch_size: int = 16
```



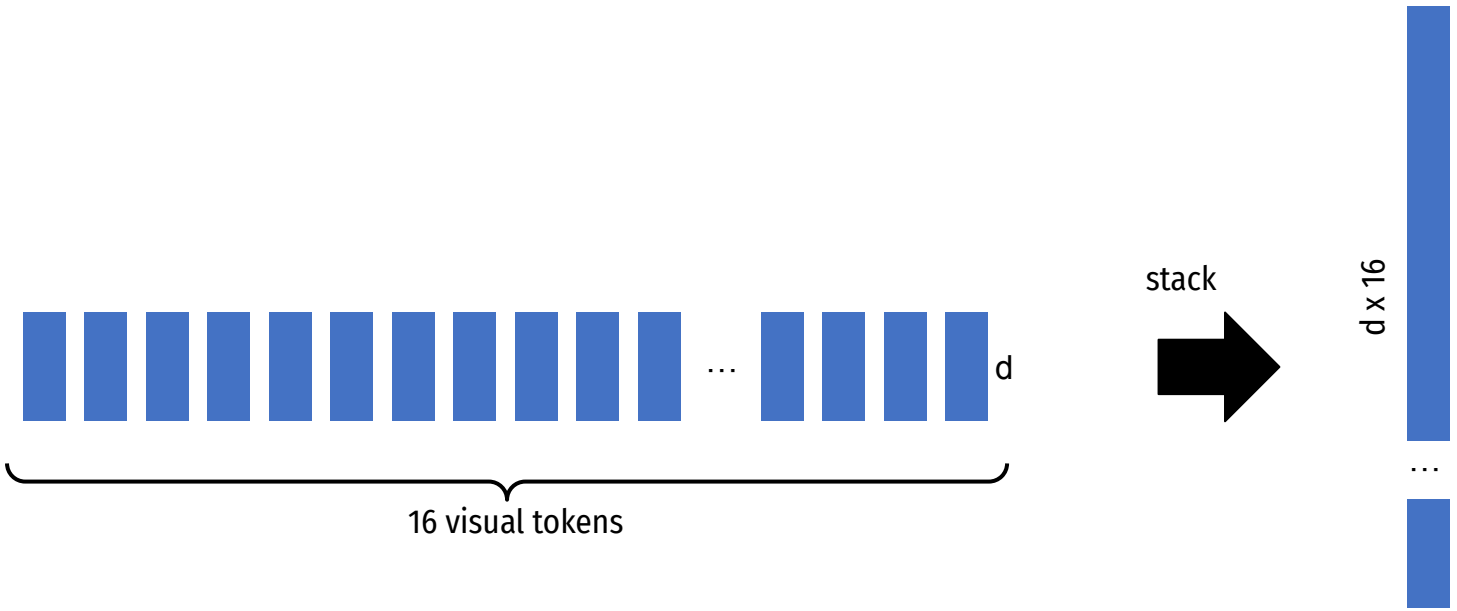
ModalityProjector

mp_pixel_shuffle_factor: 4
mp_image_token_length: 64



ModalityProjector

mp_pixel_shuffle_factor: 4
mp_image_token_length: 64



ModalityProjector

mp_pixel_shuffle_factor: 4
mp_image_token_length: 64

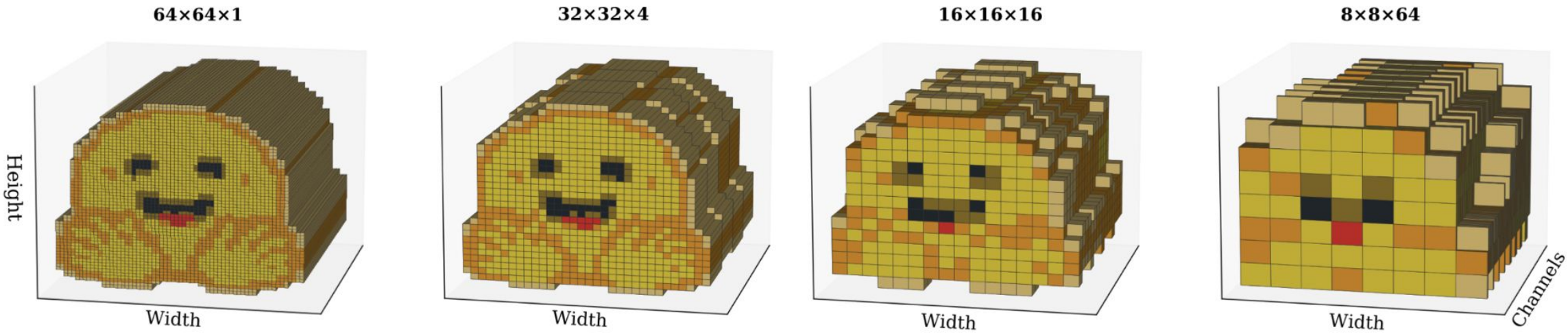
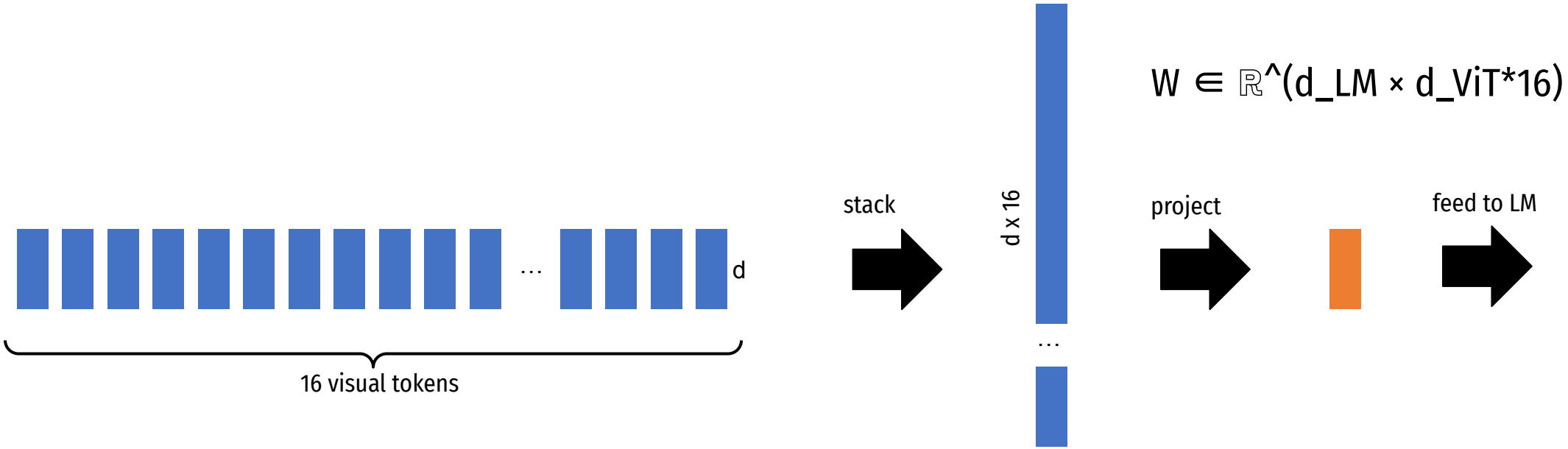


Figure 4 | Pixel shuffle. Rearranges encoded images, trading spatial resolution for increased channel depth. This reduces visual token count while preserving information density.

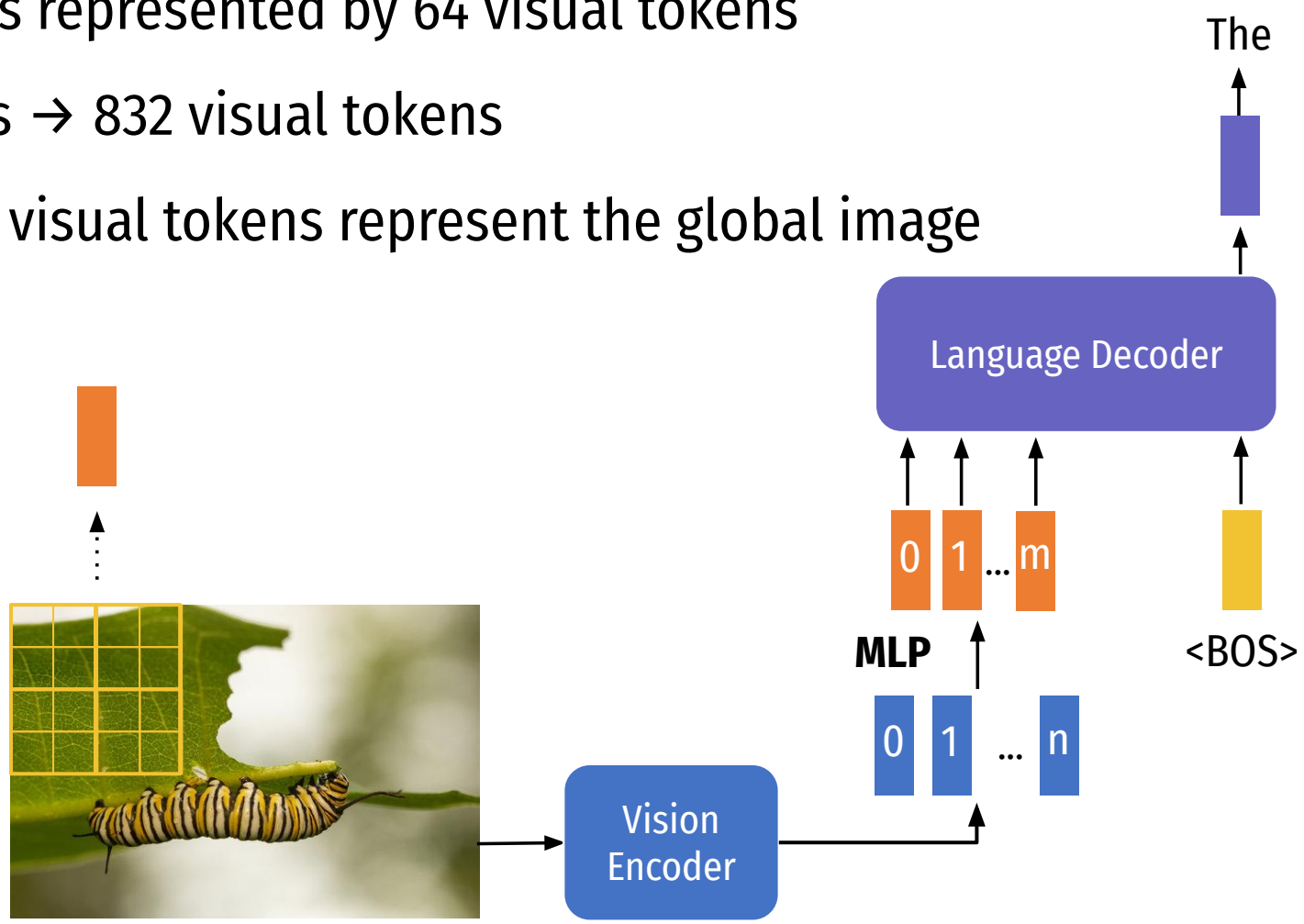
ModalityProjector

mp_pixel_shuffle_factor: 4
mp_image_token_length: 64



nanoVLM

- each split is represented by 64 visual tokens
- for 13 splits → 832 visual tokens
- the first 64 visual tokens represent the global image



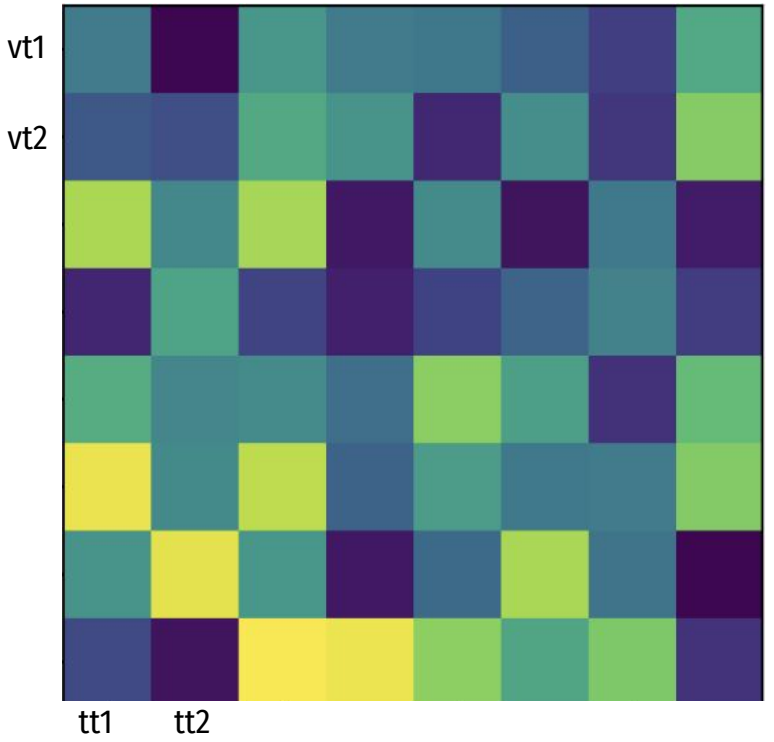
Lab 3:

- In-class: plot the attention of head *4* on layer *11* from every text token in the question to every vision token in the global image.
  - deliverable: a heatmap

Image: link

Query: What is the name of the dog's toy?

Note: each row in the full attention matrix should sum up to 1.



Lab 4:

- Implement logit lens on the image tokens from layer 16 and 32
  - deliverable: two lists of tokens (one for each layer) for this image

Infrastructure for final evaluation:

- RunPod A40