



# Deep Learning Toolkit (*Einops*)

Rowel Atienza, PhD

University of the Philippines

[github.com/roatienza](https://github.com/roatienza)

2022

# Outline

Environment, Code Editor

Python

Tensor libraries – numpy, einsum, **einops**

PyTorch, Timm

Huggingface (HF), Gradio

HF Accelerator, GitHub

Pytorch Lightning

# Einstein Operations: Einops

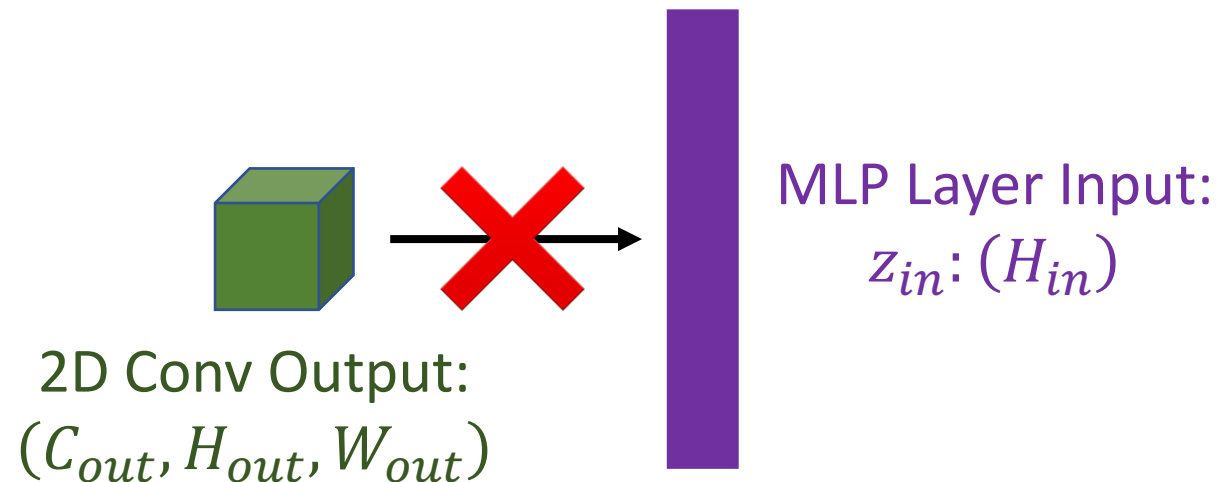
Rogozhnikov, Alex. "Einops: Clear and Reliable Tensor Manipulations with Einstein-like Notation." *International Conference on Learning Representations*. 2022.

<https://github.com/arogozhnikov/einops>

# Motivation: Tensor shape and size between layers IO must match

Input/Output shape of 2D convolution:  $z_{in}: (C_{in}, H_{in}, W_{in})$  and  $z_{out}: (C_{out}, H_{out}, W_{out})$  where  $C$ : channel,  $H$ : height and  $W$ : width

Input/Output shape of an MLP layer:  $z_{in}: (H_{in})$  and  $z_{out}: (H_{out})$  where  $H$ : height



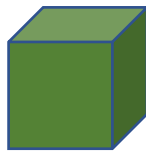
Solution: Reshape the 2D convolution output to match the input requirement of MLP

$$z_{out}: (C_{out}, H_{out}, W_{out}) \rightarrow (C_{out} * H_{out} * W_{out})$$

$$z_{in}: (H_{in})$$

$$\therefore H_{in} = C_{out} * H_{out} * W_{out}$$

2D Conv Output:  
 $(C_{out}, H_{out}, W_{out})$



Reshape



$$(C_{out} * H_{out} * W_{out})$$

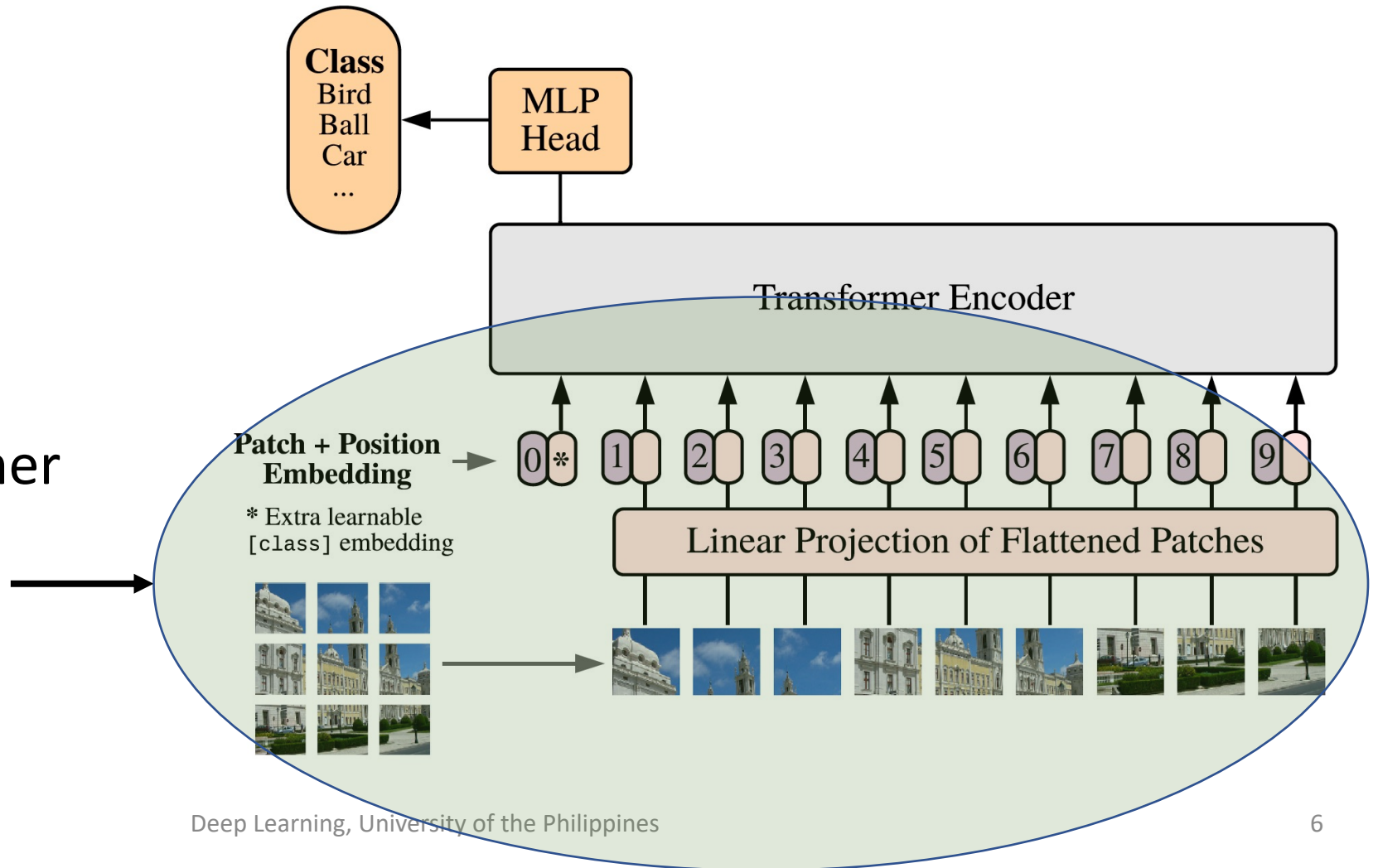


MLP Layer Input:  
 $z_{in}: (H_{in})$

# Motivation: Upsize, Downsize, Stack, Split, View, Permute, etc

## Vision Transformer (ViT)

In Vision Transformer (ViT), we split an input image into patches.



# Numpy/PyTorch vs Einops APIs

Operation	Numpy/PyTorch	Einops
Transpose	<code>transpose</code>	<code>rearrange</code>
Reshape	<code>reshape/view</code>	<code>rearrange</code>
Upsize	<code>repeat/upsample</code>	<code>repeat</code>
Downsize	<code>interpolate</code>	<code>reduce</code>
Split	<code>split</code>	<code>rearrange</code>
Permute	<code>permute</code>	<code>rearrange</code>

# Install and Import

## Install

```
pip install einops
```

## Import

```
from einops import rearrange, repeat, reduce
```



# Flatten

```
img = image.imread("aki_dog.jpg")  
img = rearrange(img, "h w c -> (h w c)")
```



(224, 224, 3)

→ (150528,)

# Syntax

Input tensor

Output shape:  
Fuse height, width,  
channel by multiplying  
them together

rearrange (img, "h w c -> (h w c) ")

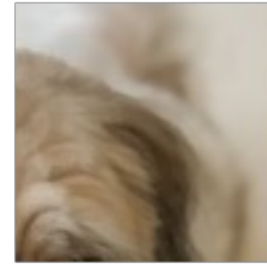
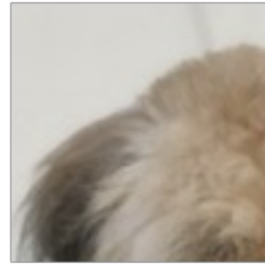
Input shape:  
height, width, channel

The diagram illustrates the syntax of the `rearrange` function. It shows the function call `rearrange (img, "h w c -> (h w c) ")`. Above the `img` parameter, a bracket indicates it is the 'Input tensor'. Below the shape string, a bracket under 'h w c' indicates the 'Input shape: height, width, channel'. Another bracket above '(h w c)' in the output string indicates the 'Output shape: Fuse height, width, channel by multiplying them together'.

# Flatten in numpy

```
img = np.reshape(img, (-1,))
```

# Image to Patches

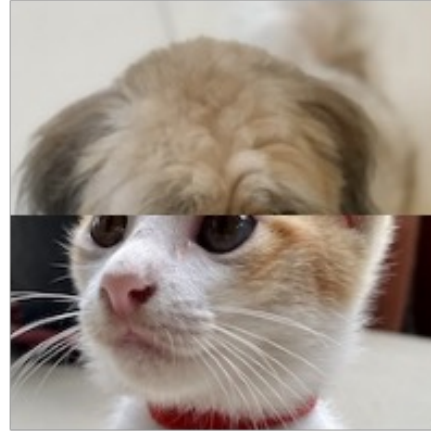


`rearrange(img, "(p1 h) (p2 w) c -> (p1 p2) h w c", p1=2, p2=2)`

$\underbrace{\hspace{1.5cm}}$   $\underbrace{\hspace{1.5cm}}$   $\underbrace{\hspace{1.5cm}}$

Split the height into 2      Split the width into 2      An array of 4 image patches

# Mixing 2 images



# Load and store

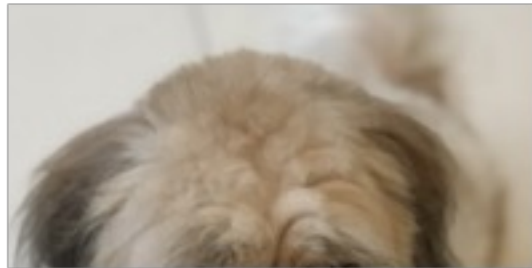
```
img1 = image.imread("aki_dog.jpg")  
img2 = image.imread("wonder_cat.jpg")  
imgs = np.array([img1, img2])
```

**Shape is now (b h w c) where batch size  
b=2**



# Create a 2D array of images

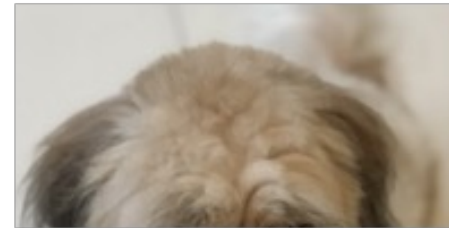
```
imgs = rearrange(imgs, "b (k h) w c -> k b h w c", k = 2)
```



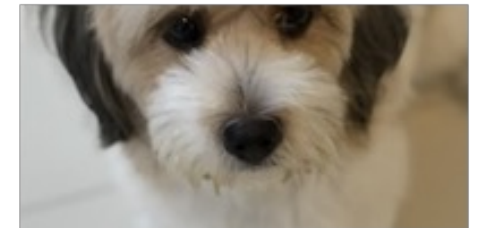
# Reverse the order of the lower halves

```
imgs = np.concatenate([imgs[::2], imgs[1::2,::-1]], axis=0)
```

Upper  
half  
array



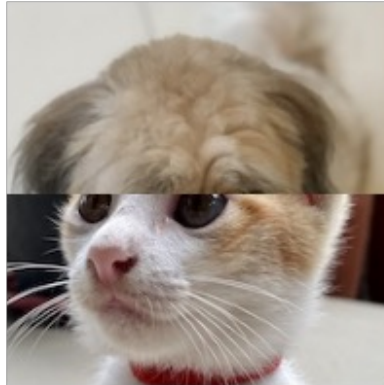
Lower  
half  
array





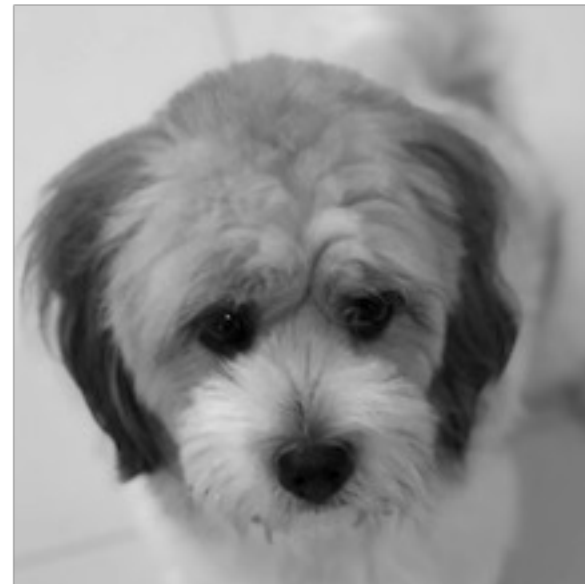
# Lastly, we fuse the 2D array into 1D array of images

```
imgs = rearrange(imgs, "i j h w c -> j (i h) w c")
```



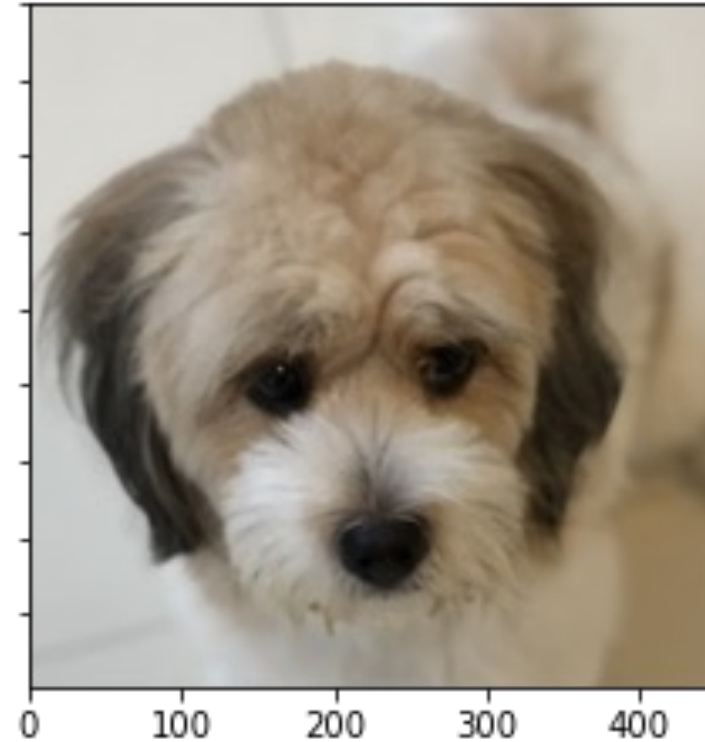
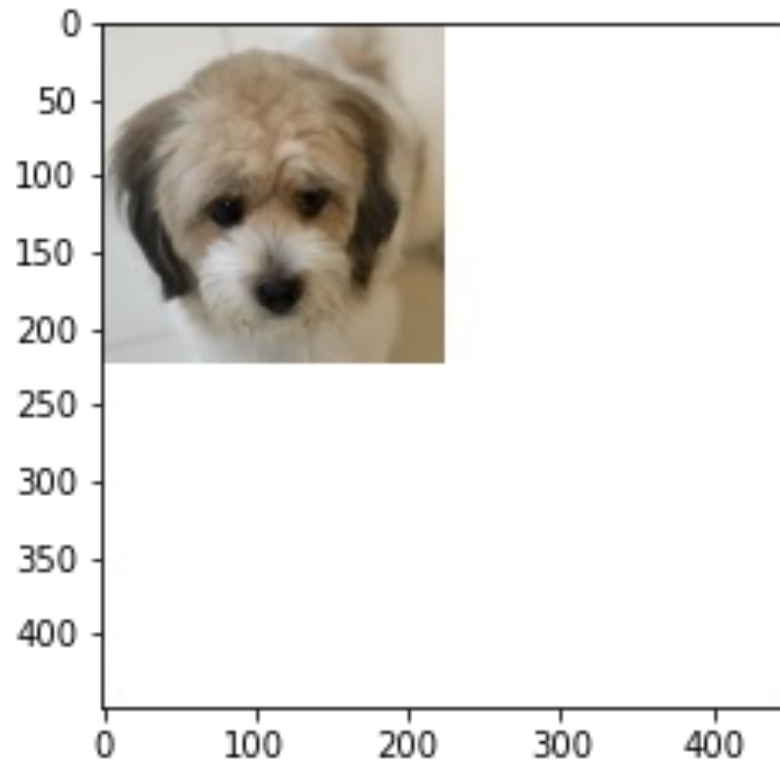
# RGB to Grayscale

```
img = reduce(img, "h w c -> h w", 'mean')
```



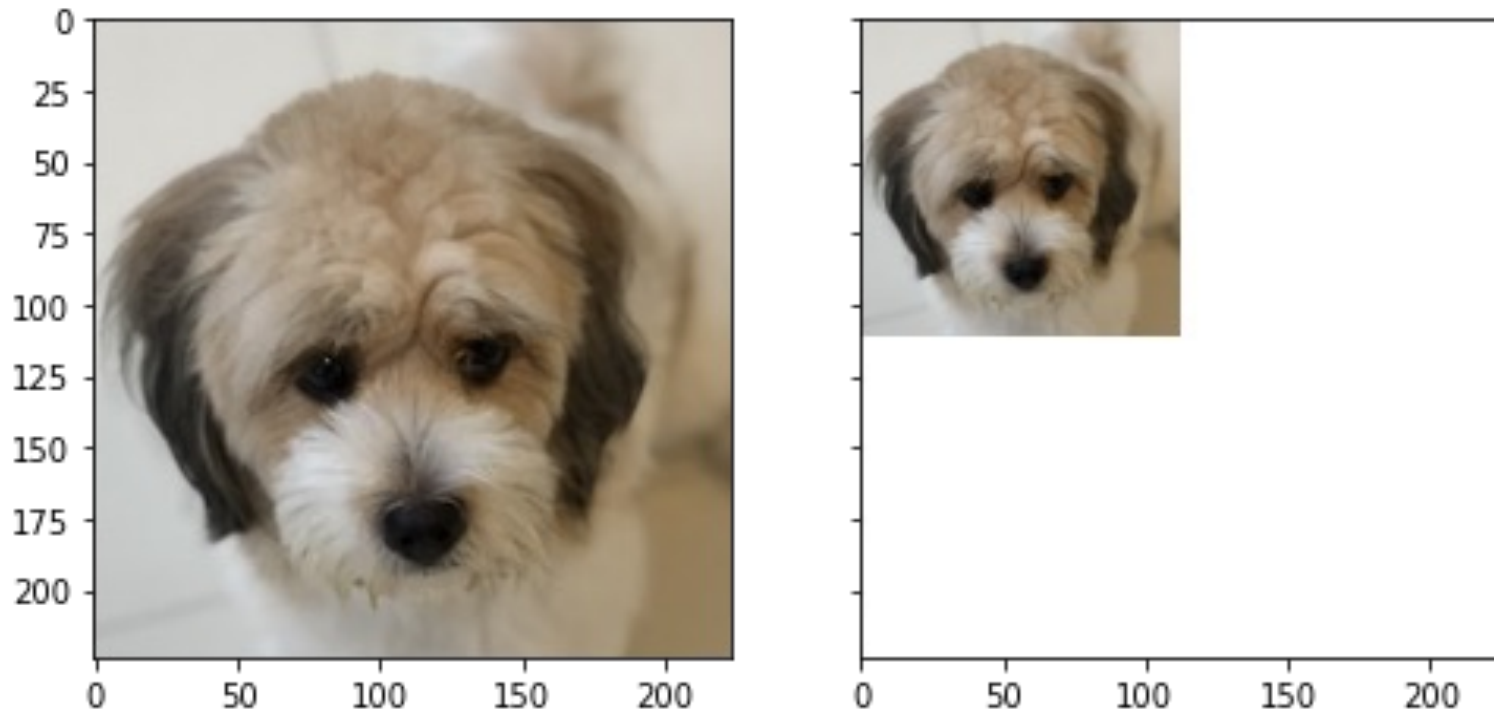
# Upsize

```
repeat(img, "h w c -> (h 2) (w 2) c")
```



# Downsize

```
reduce(img, "(h 2) (w 2) c -> h w c", 'mean')
```



# End

[https://github.com/roatienza/Deep-Learning-Experiments/blob/master/versions/2022/tools/python/einops\\_demo.ipynb](https://github.com/roatienza/Deep-Learning-Experiments/blob/master/versions/2022/tools/python/einops_demo.ipynb)