# Generative AI with Diffusion Models

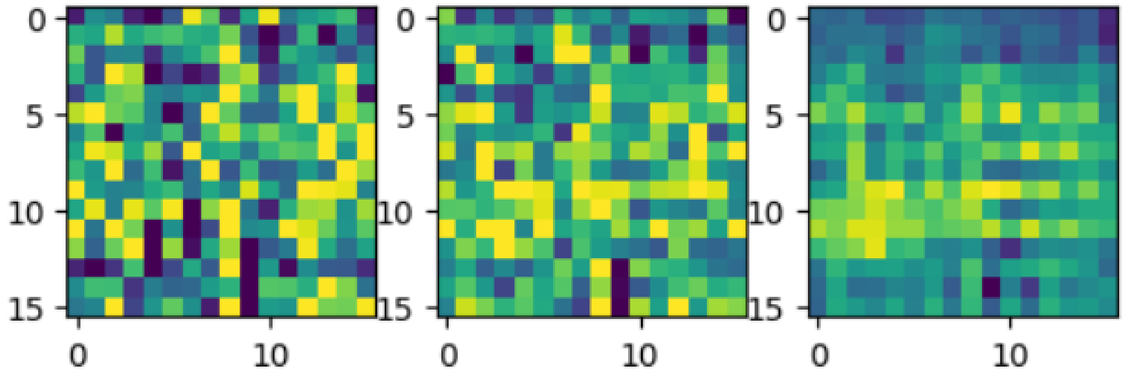
Part 3: Optimizations

# Agenda

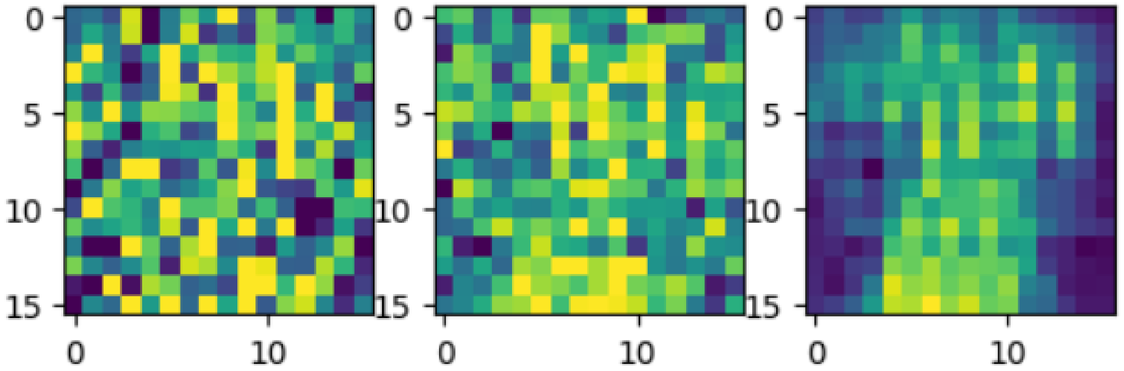
- Part 1: From U-Nets to Diffusion
- Part 2: Denoising Diffusion Probabilistic Models
- Part 3: Optimizations
- Part 4: Classifier-Free Diffusion Guidance
- Part 5: CLIP
- Part 6: Wrap-up & Assessment

# The Checkerboard Problem

# The Checkerboard Problem




Shoe?

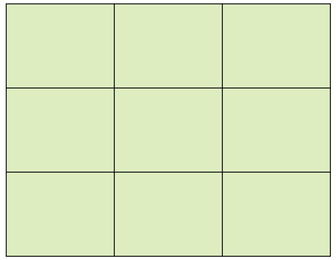



Shirt?

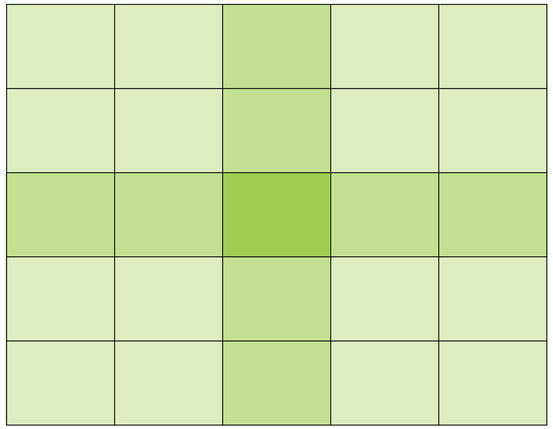
# The Checkerboard Problem

## The Impact of Stride

Kernel Size = 3



Stride = 2

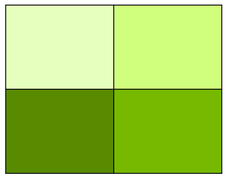



A cozy green plaid blanket, fairy tale drawing

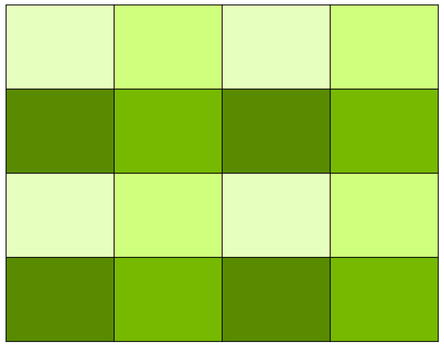
# The Checkerboard Problem

## The Impact of Stride

Kernel Size = 2



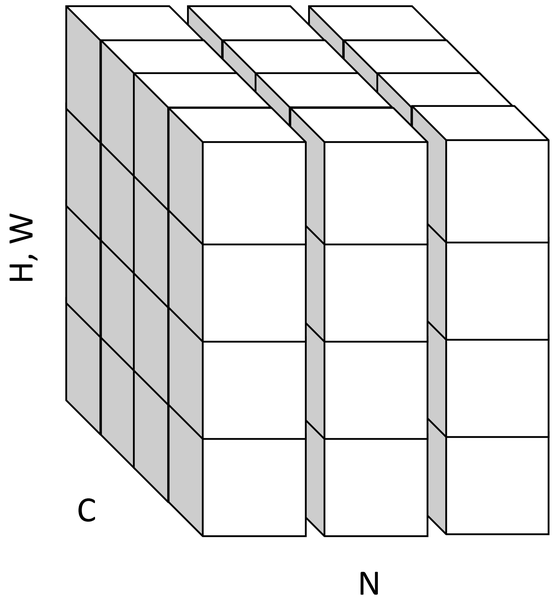
Stride = 2



# Group Normalization

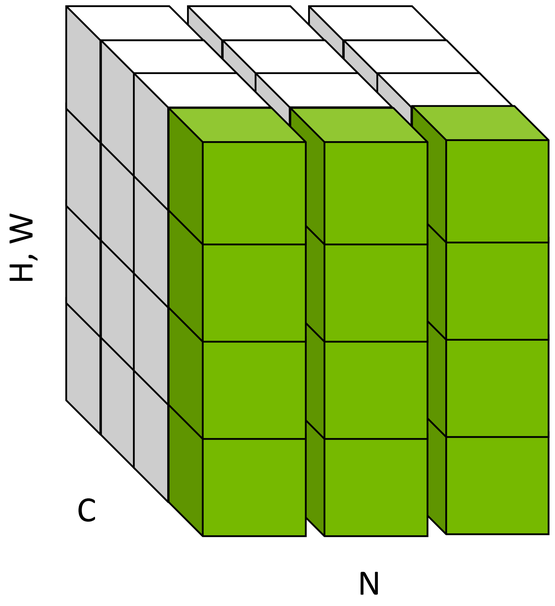
# Group Normalization

## Batch Normalization Review



# Group Normalization

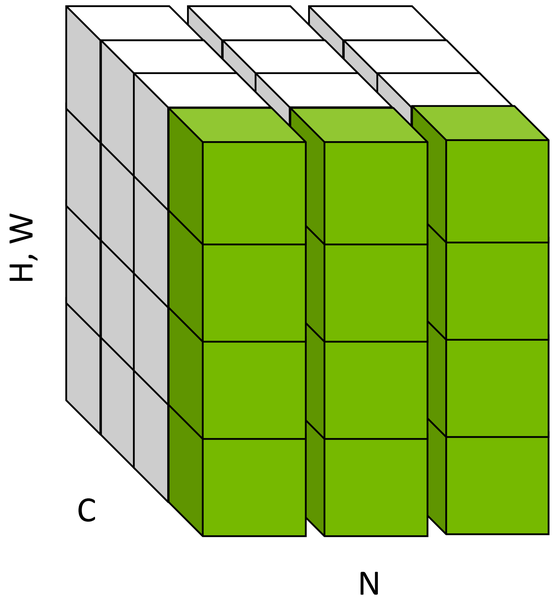
## Batch Normalization Review



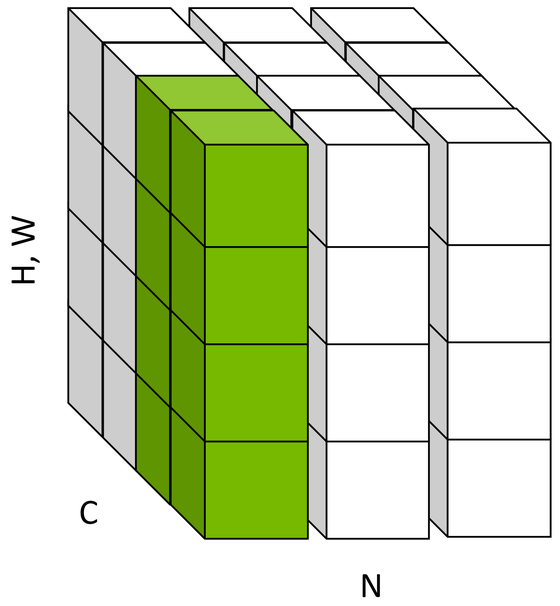

Batch Normalization

# Group Normalization

## Types of Group Normalization



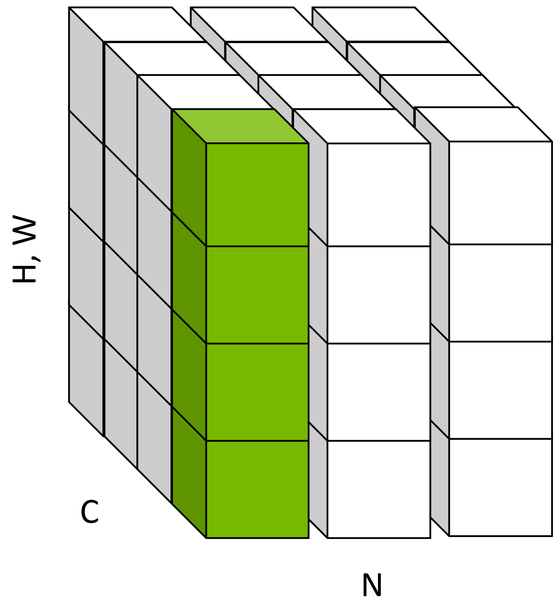

Batch Normalization




Group Normalization

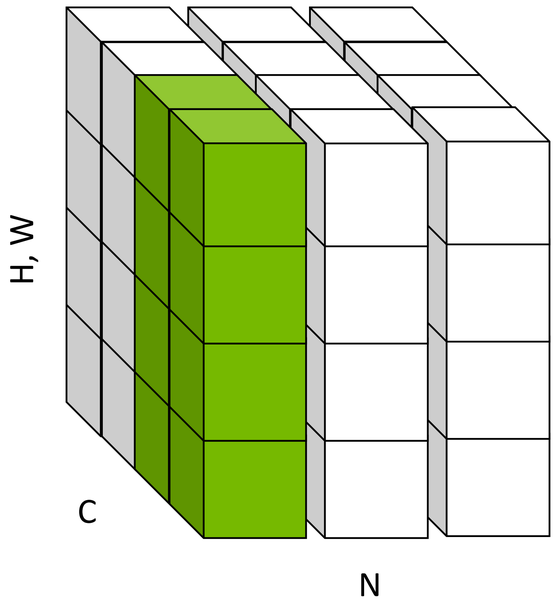
# Group Normalization

## Types of Group Normalization



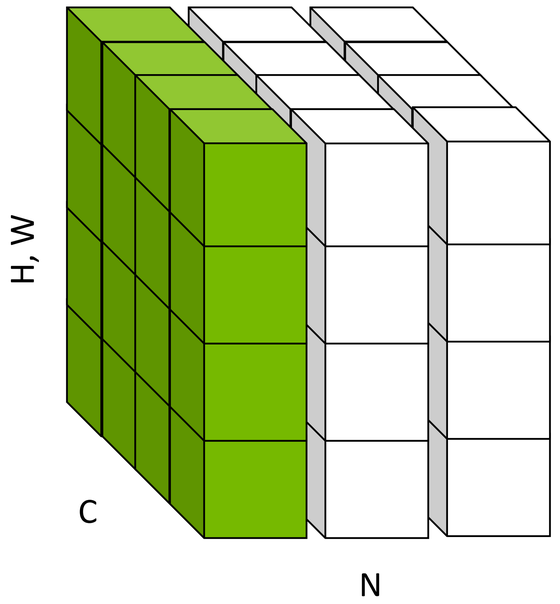

Instance Normalization

Group Size = 1




Group Normalization

Group Size = 2

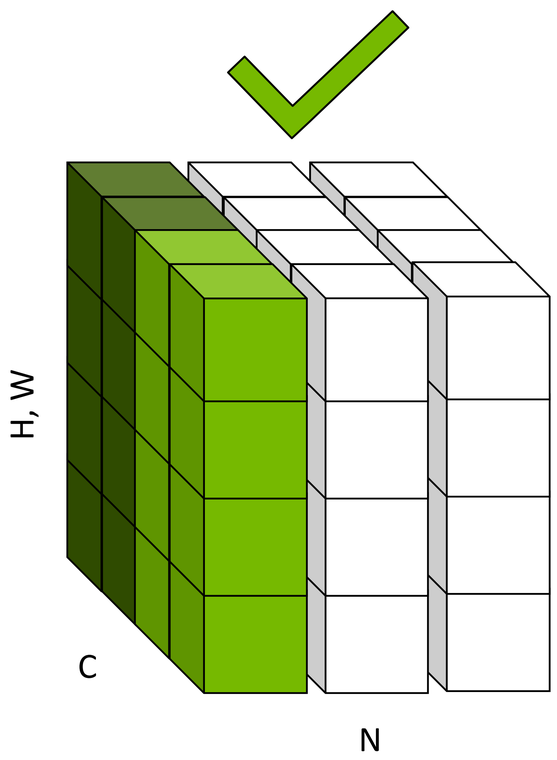



Layer Normalization

Group Size = All

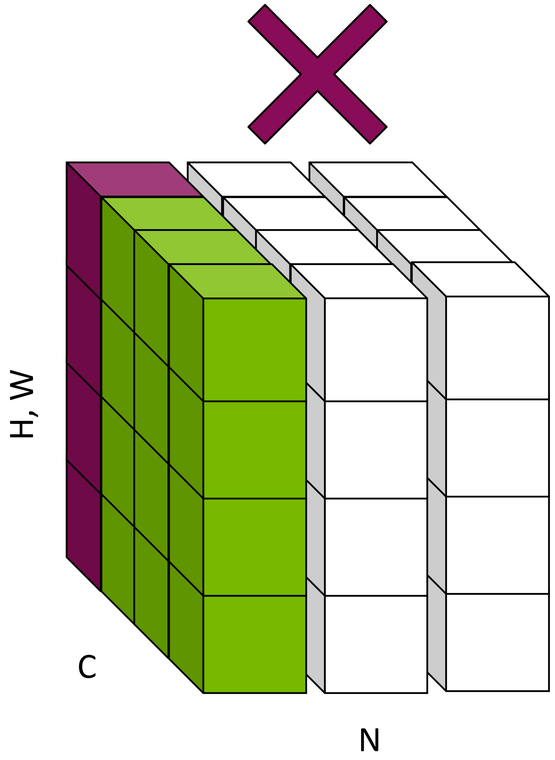
# Group Normalization

## Types of Group Normalization




Group Normalization

Group Size = 2



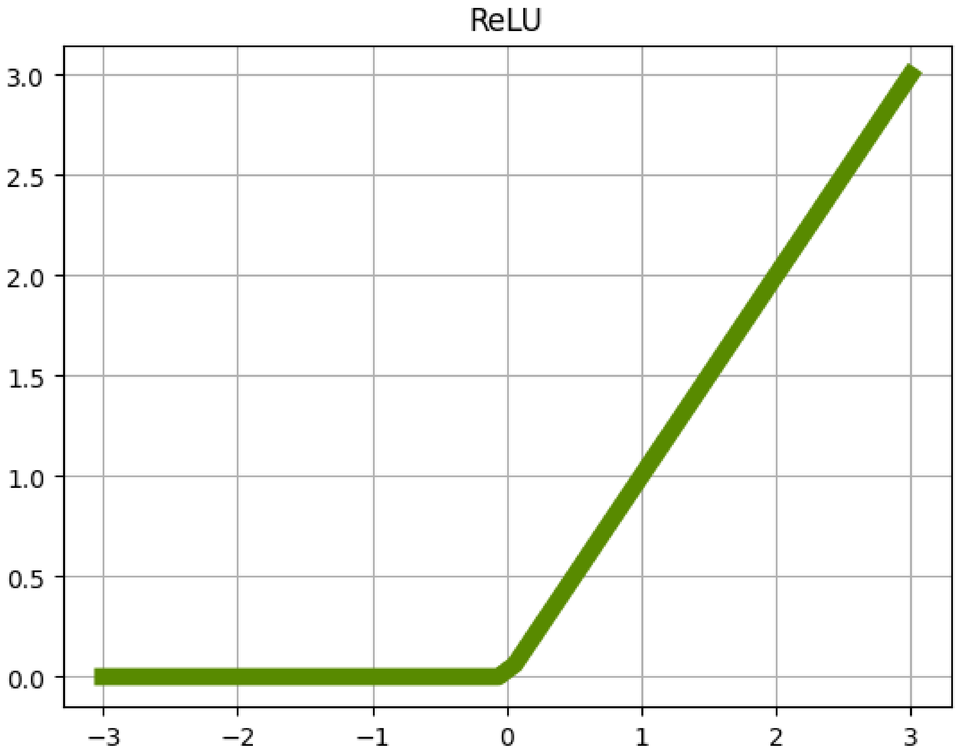

Group Normalization

Group Size = 3

# Gaussian Error Linear Unit (GELU)

# GELU

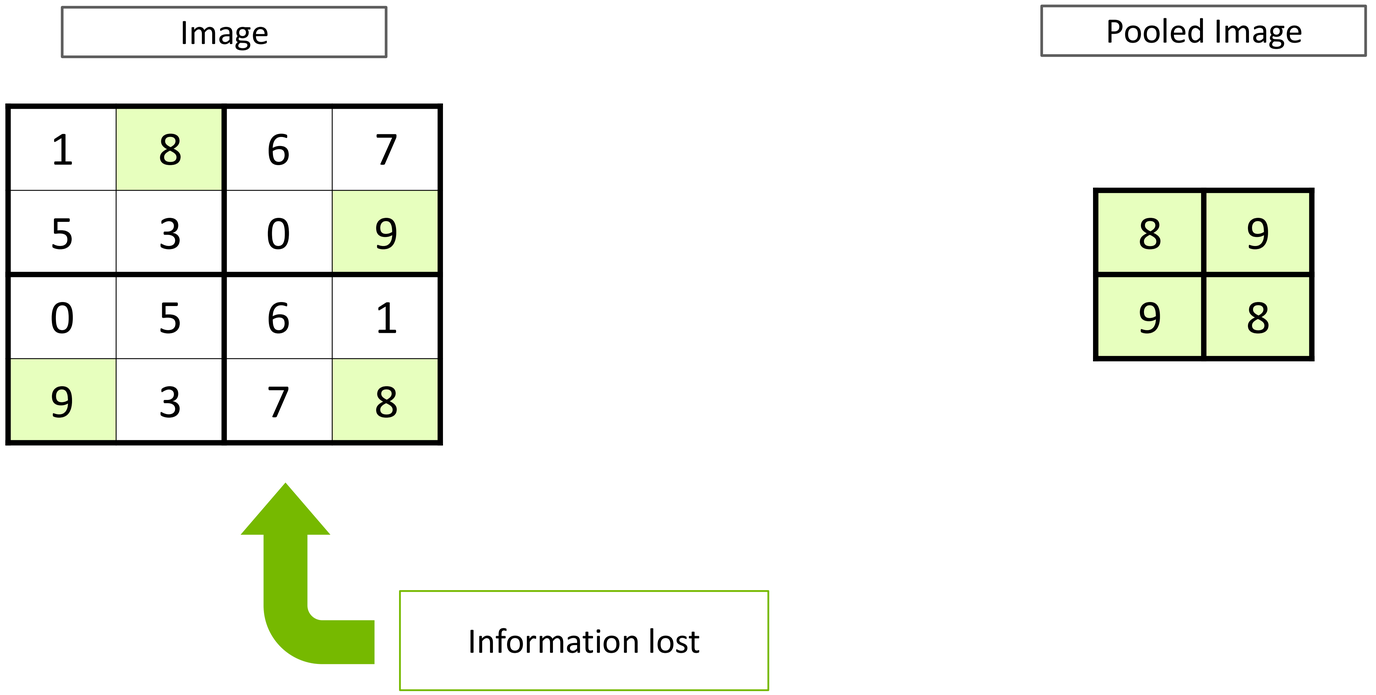
Gaussian Error Linear Unit




# Rearrange Pooling

# Pooling

## Max Pooling Review

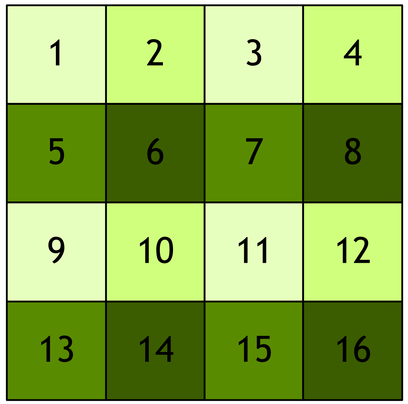


# Einops

## Better Dimension Manipulation

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

Cut image into strips and stack

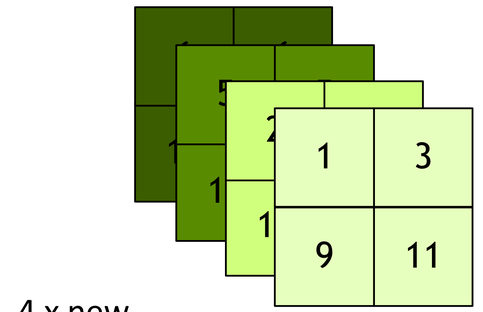


# Einops

## Better Dimension Manipulation

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

Cut image into strips and stack




4 x new Channels

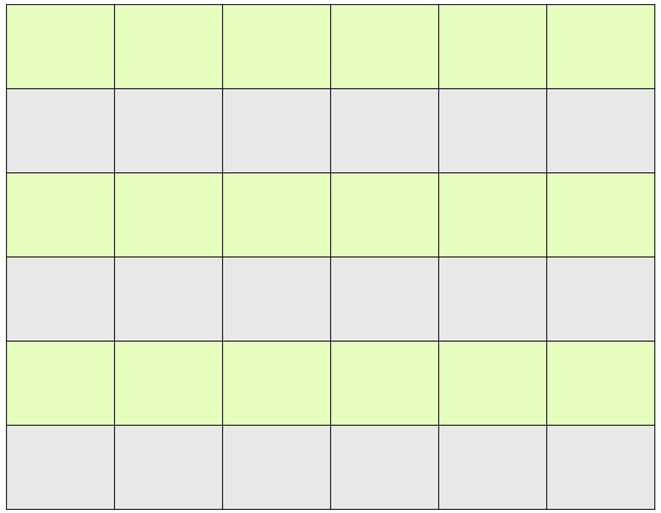
$^1/_2$ Height

$^1/_2$ Width

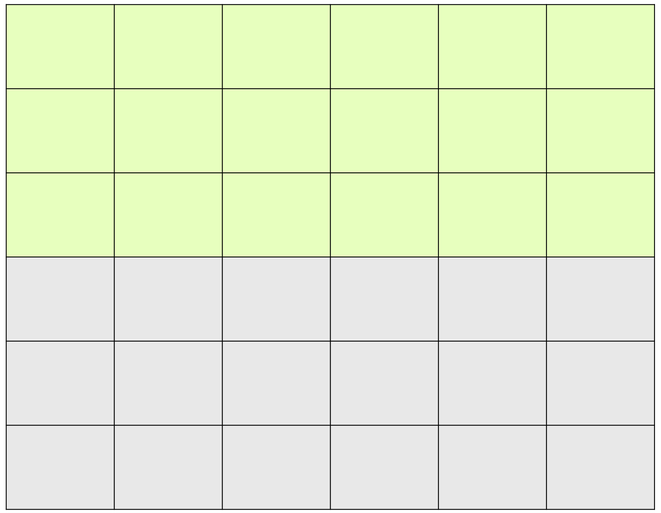
# Einops

## Order Matters

```
Rearrange(
  "c (h p1) w -> (c p1) h w", p1=2
)
```



```
Rearrange(
  "c (p1 h) w -> (c p1) h w", p1=2
)
```



# Sinusoidal Position Embeddings

# Time as a Sequence

## How to Represent Time as Discrete Steps?

t

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | … |
|---|---|---|---|---|---|---|---|---|---|---|

As a one-hot encoding?

t = 7

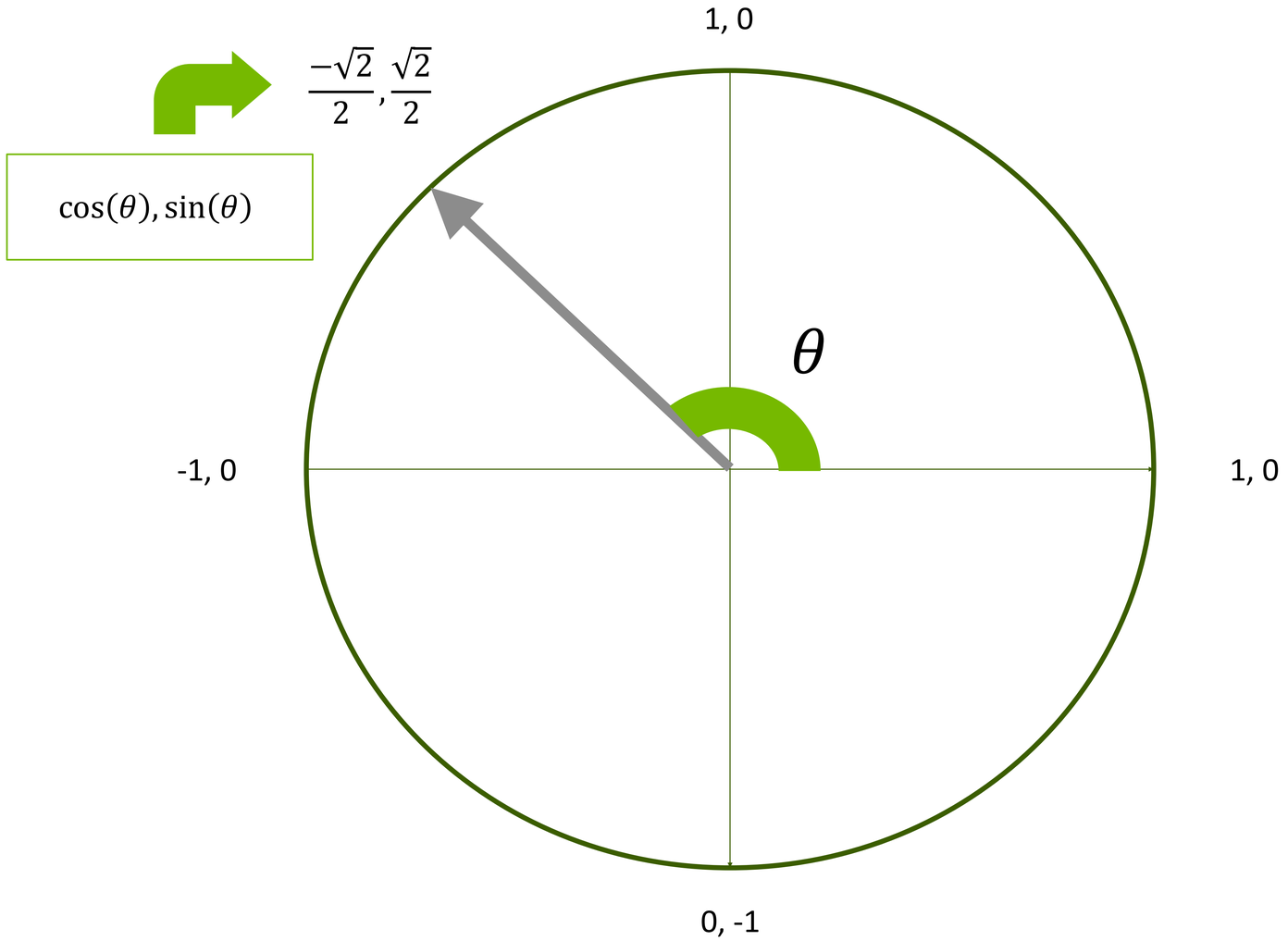
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | … |
|---|---|---|---|---|---|---|---|---|---|---|

As binary?

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | … |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | … |
| 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | … |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | … |

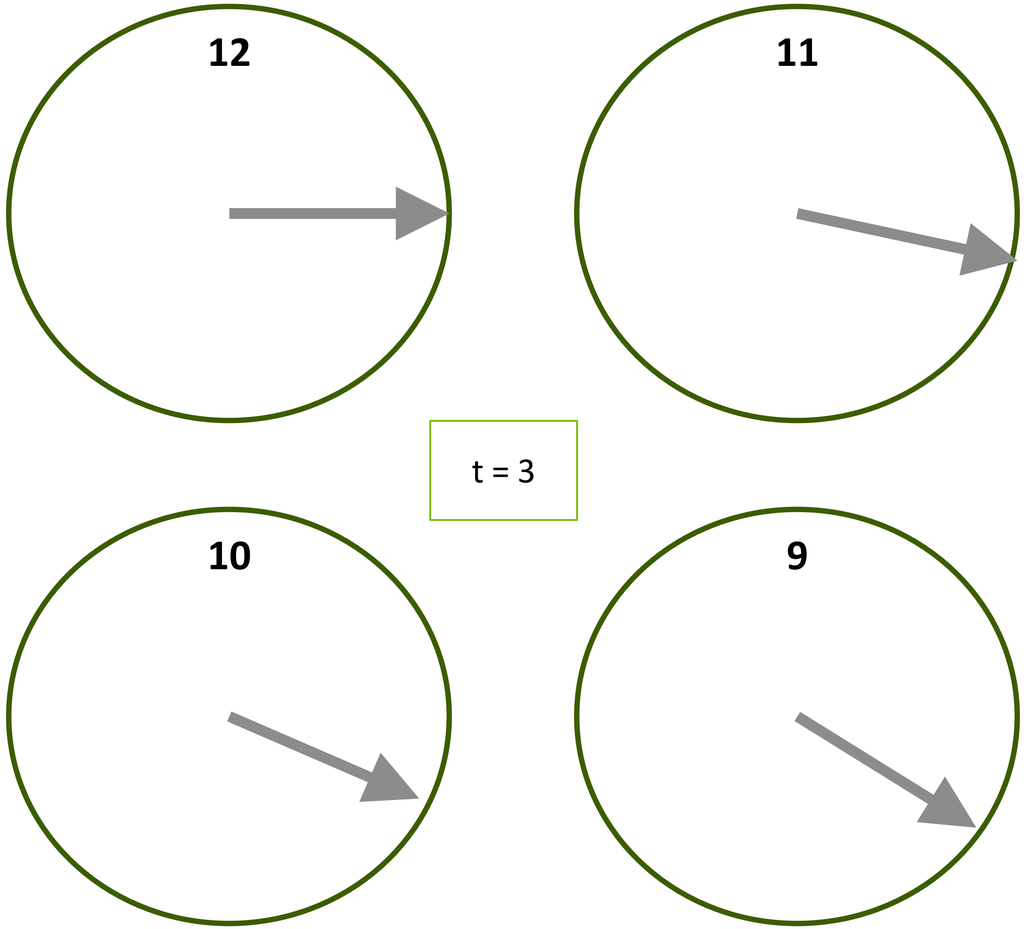
# Time as a Sequence

## As a Unit Circle?



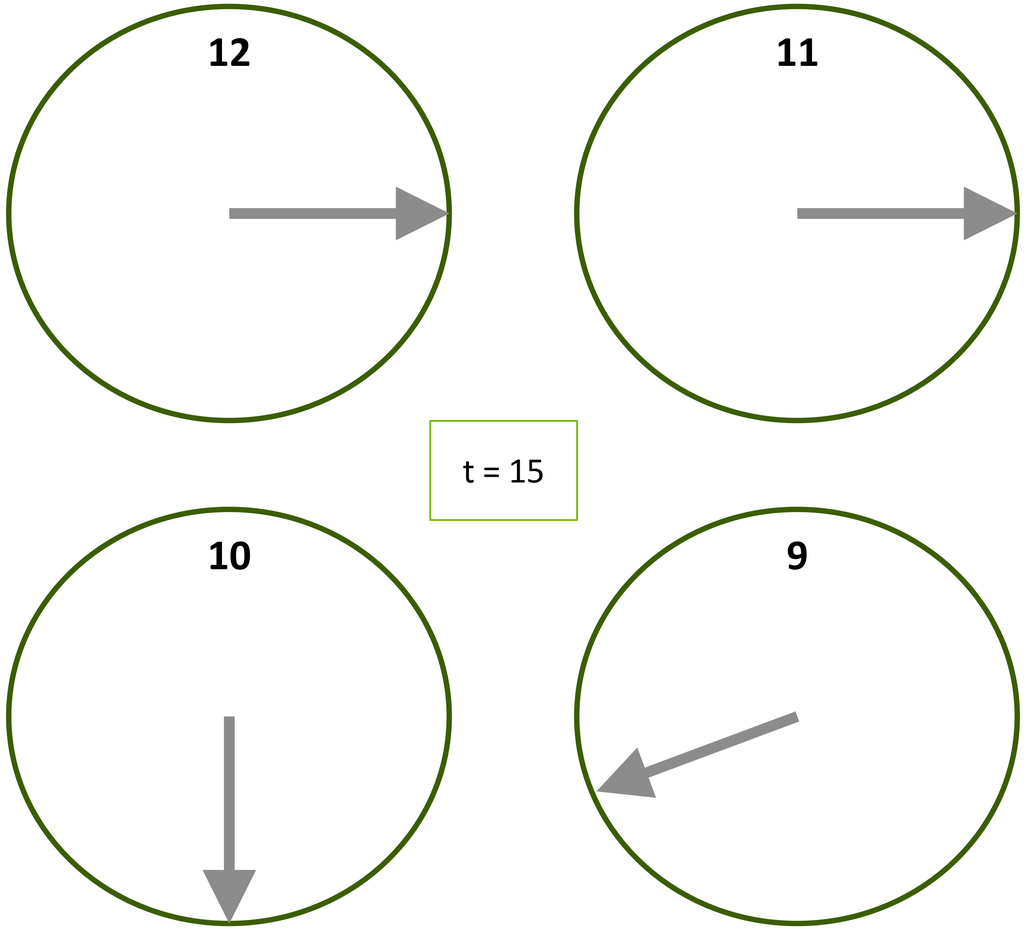
# Time as a Sequence

## As Unit Circles?



# Time as a Sequence

## As Unit Circles?



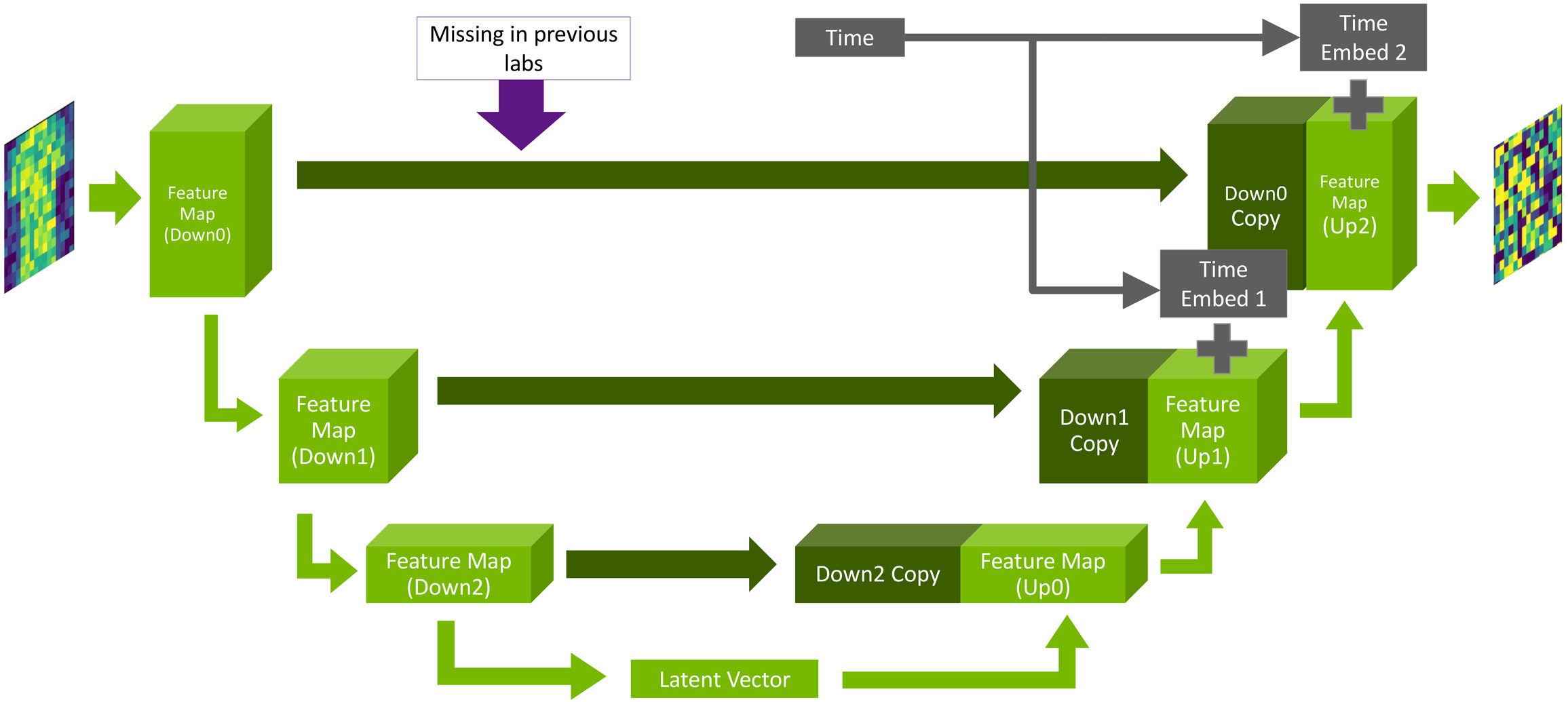


A bunch of abstract clocks with different numbers on them

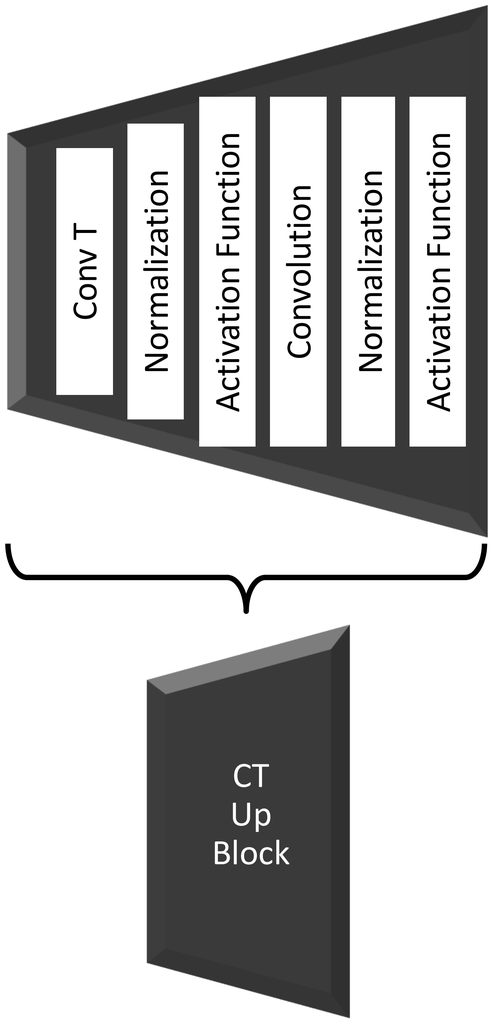
# Deeper Networks

# The U-Net Architecture

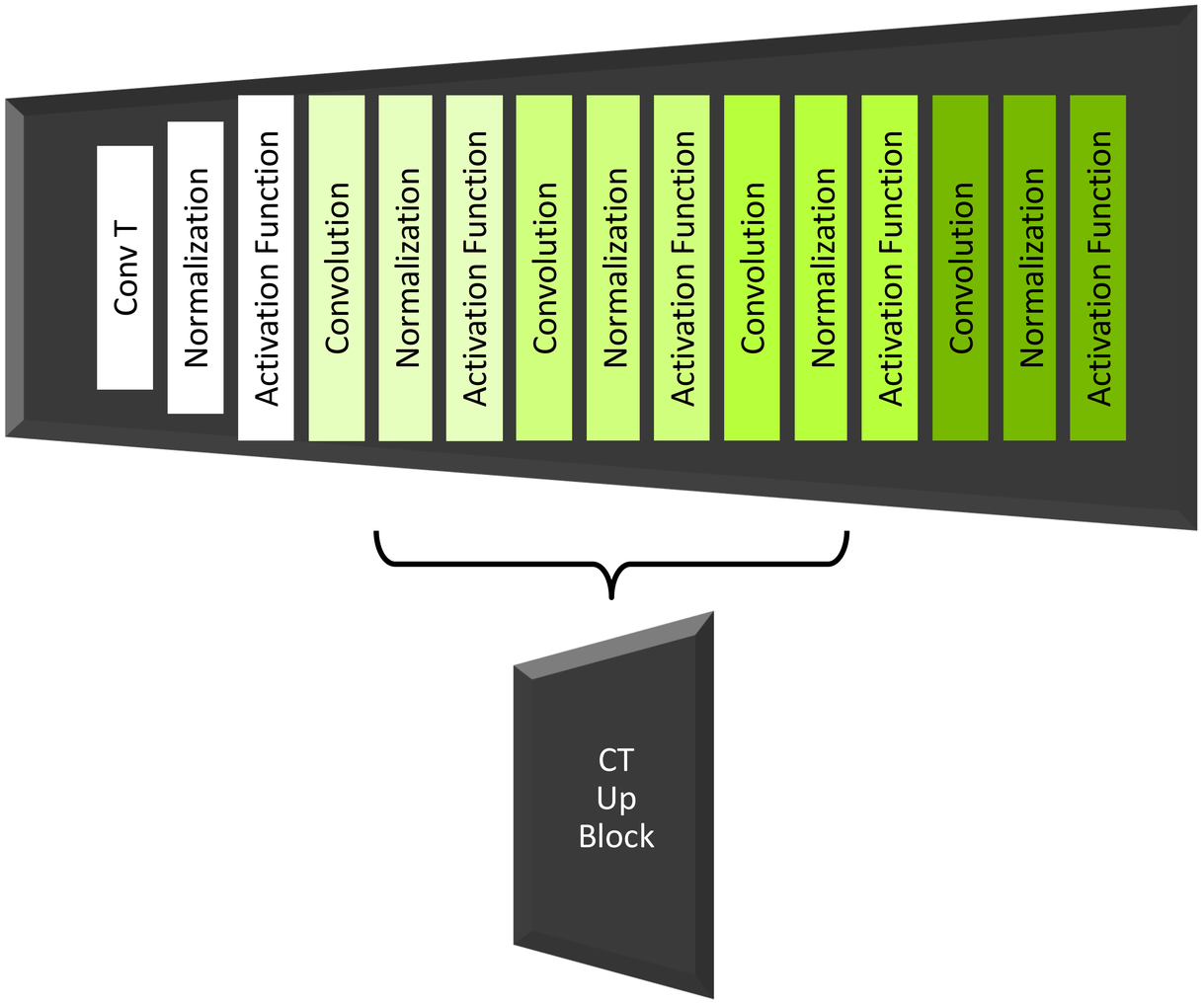
## Adding Time



# Adding Depth



# Adding Depth



# Let’s get started!