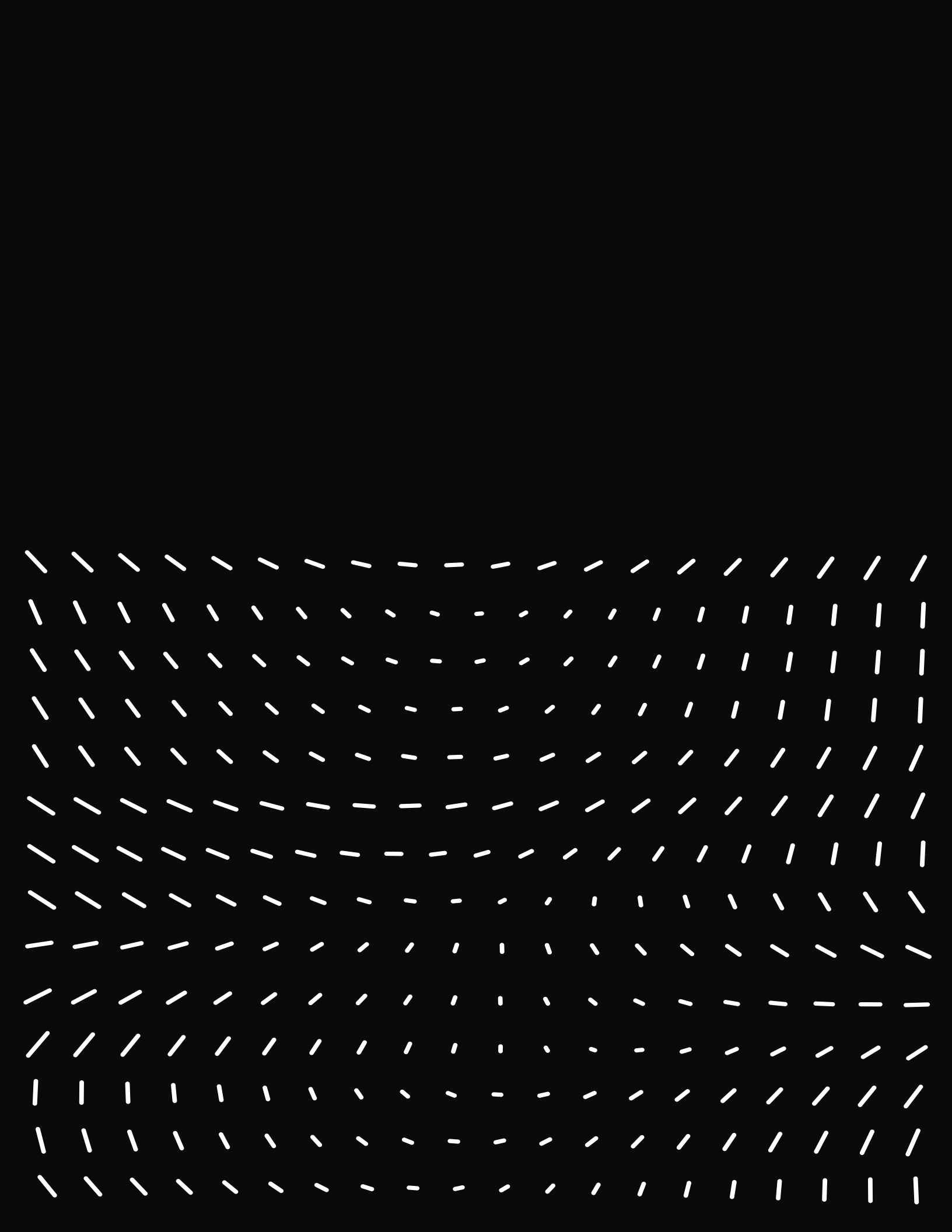
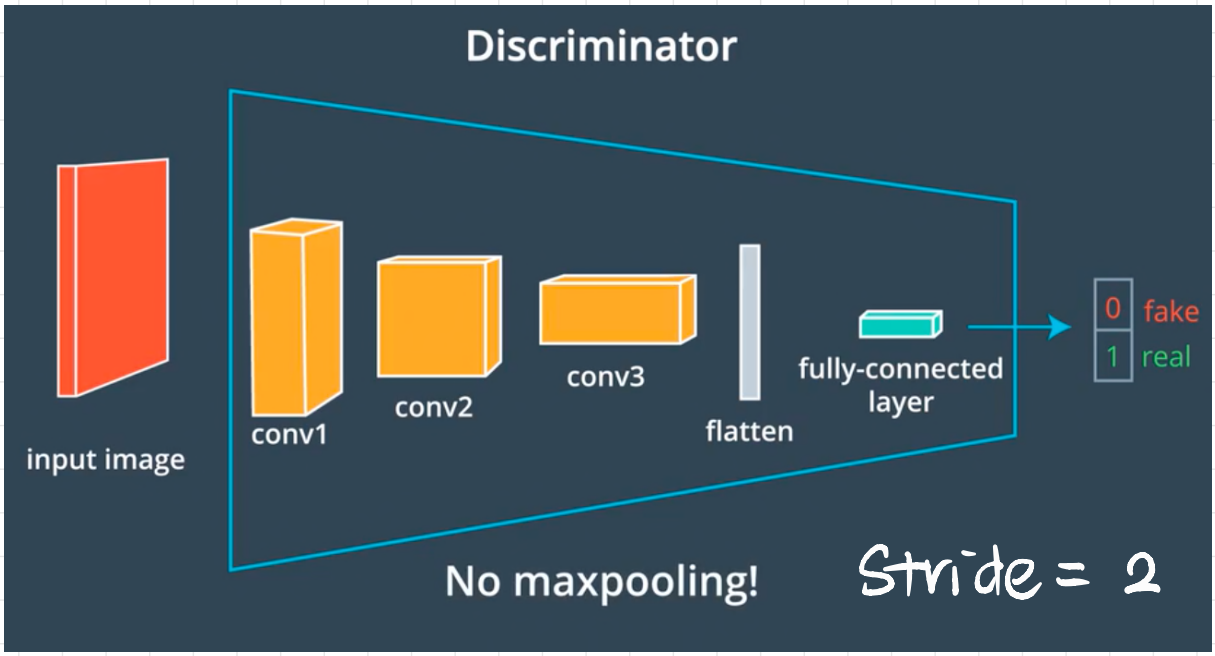


Discriminator



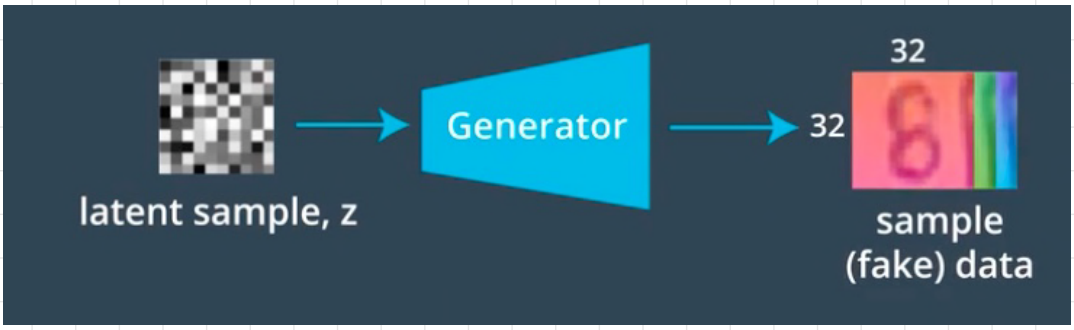
Stride=2: Downsampling dim by 2.

All hidden layers have **Batch Norm** & **Leaky ReLU** implemented in.

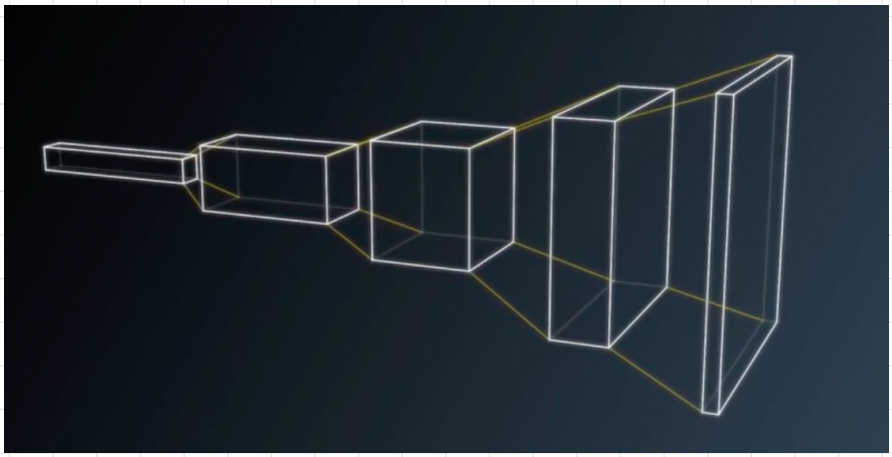
Batch Norm: scale outputs to have mean=0 & variance = 1.

Leaky ReLU: Multiply any negative number by a negative slope

Generator:



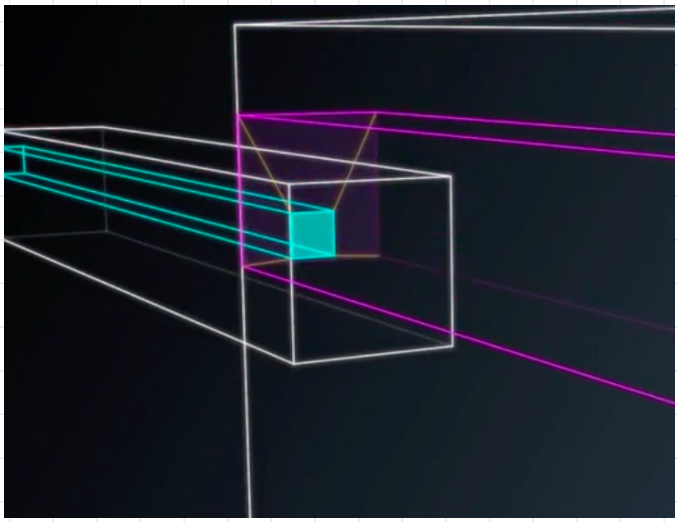
Need to upsample output to be the same shape of the input images by **Transverse Conv.**



Deep & Long vector

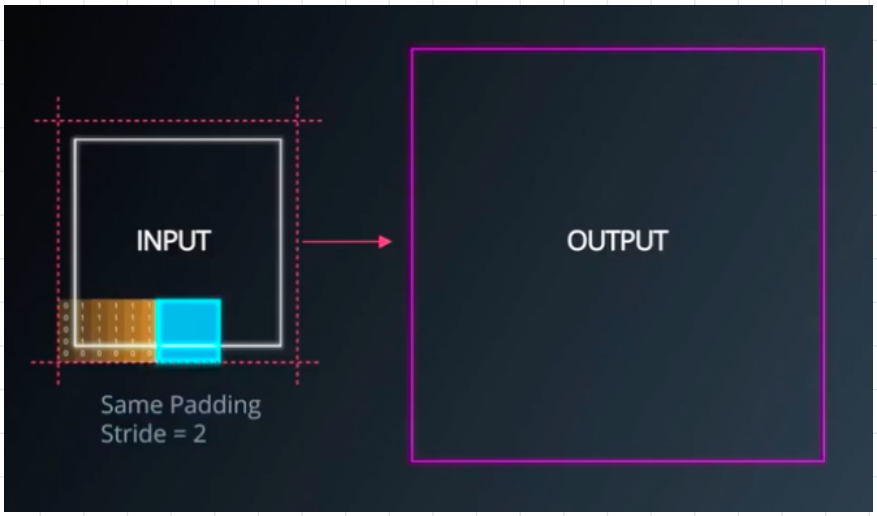
↓

Shallow image



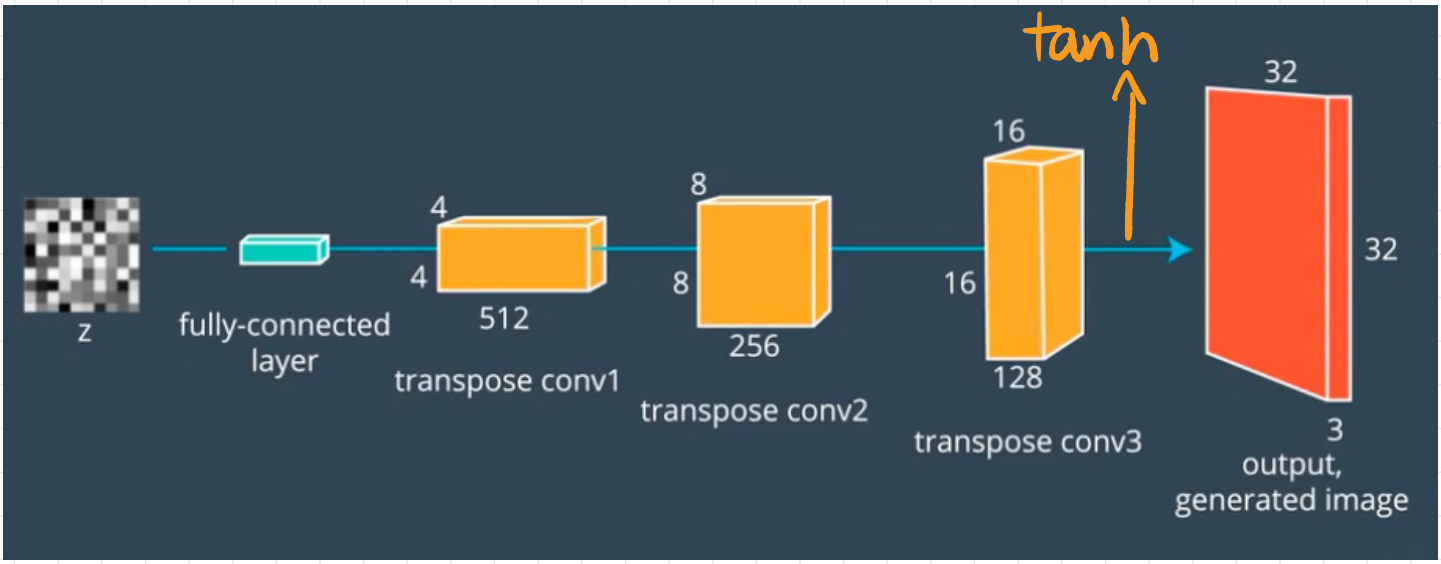
Stride = 2

Move conv kernel 1 pixel in input layer, kernel will move 2 pixels in the output layer.



Output layer size depends on the kernel size.

With a stride of 2, the output dim doubles.



Batch Norm but **ReLU** on all hidden layers

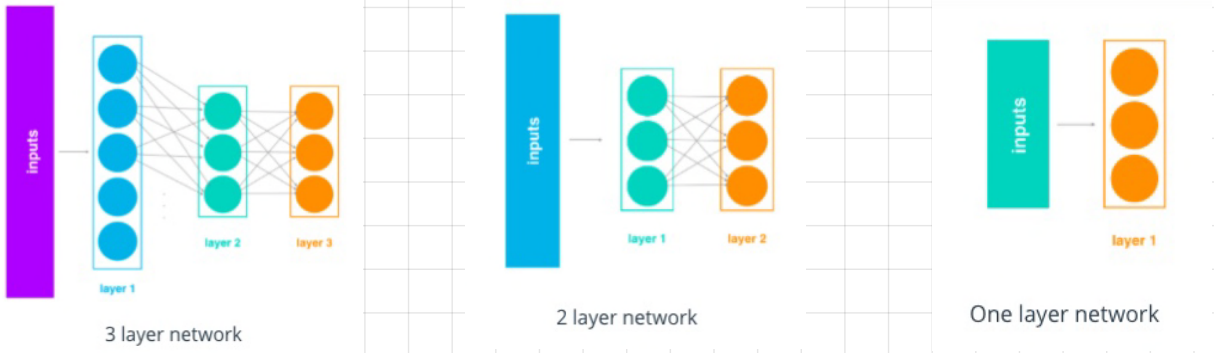
tanh on the last layer, giving us a value for the output in a range $[-1, 1]$.

Batch Normalization

During training, we normalize each layer's inputs by using the mean & std of the values in current batch.

> *Specifically, batch normalization normalizes the output of a previous layer by* ***subtracting the batch mean and dividing by the batch standard deviation****.*

Normalization @ every layer



Think of the output of the current layer as the input of the proceeding layer.

Outputs $\Longleftrightarrow$ inputs

Batch Norm = Norm at every layer/sub-network.

Internal Covariance Shift

> *In this case, internal covariate shift refers to the change in the distribution of the inputs to different layers. It turns out that training a network is most efficient when the distribution of inputs to each layer is similar!*

Math.

Average value for the batch:

The avg. value coming out of any particular layer before we pass it through its non-linear activation function & then feed it as an input to the next.

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta$$

↓

Final batch-normalized output

- Add Batch Norm before applying the activation function!
- Turn off bias when Batch Norm.

DCGANs, SVHN Notebook

Conv. arithmetic: $o = \frac{i - k + 2p}{s} + 1$

Transverse Conv: $o = s(i-1) + a + k - 2p$

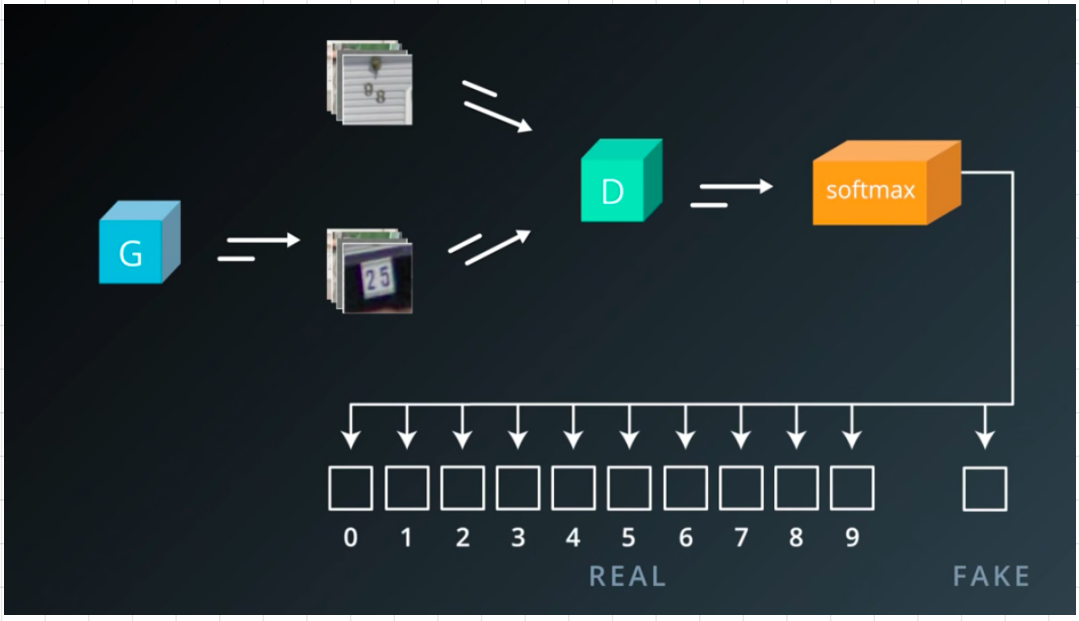
Semi-supervised Learning

- Improving a classifier.
- Real life examples aren't always labeled.
- Easier & cheaper to obtain unlabeled data!

Normally,

Train both, throw away D at the end.
- D is used to train G.

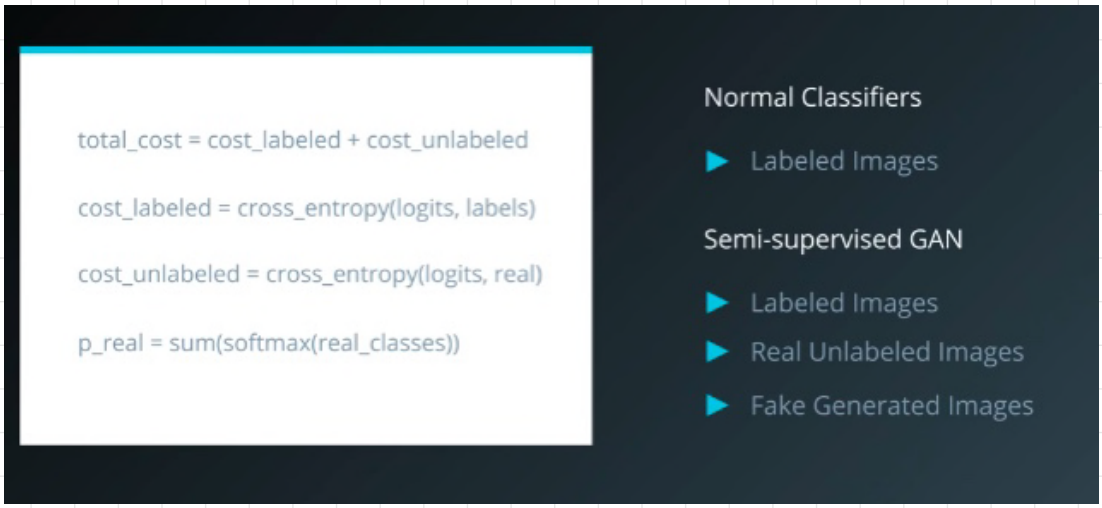
But for semi-supervised learning:



Use D as a **classifier**. classifying Real or fake.

Use **Softmax** to classify diff classes of real samples

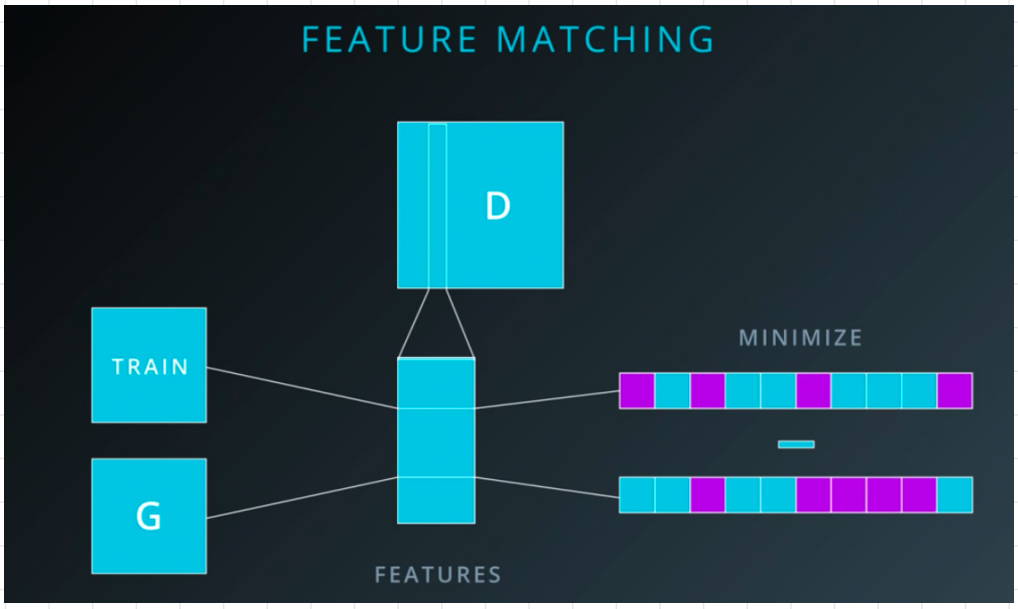
Total cost = cost_labeled + cost_unlabeled



↓ test error due to ↑ diverse samples

Feature Matching

* Needed to make the network work well!



Checkout Git Repo

Sleepychord / ImprovedGAN-pytorch.

DCGANs Workflow

Preprocessing

- Create a DataLoader & Visualize data
  - Dataset → ImageFolder → imshow
- Rescale feature (-1, 1)

Define the Model

- Discriminator