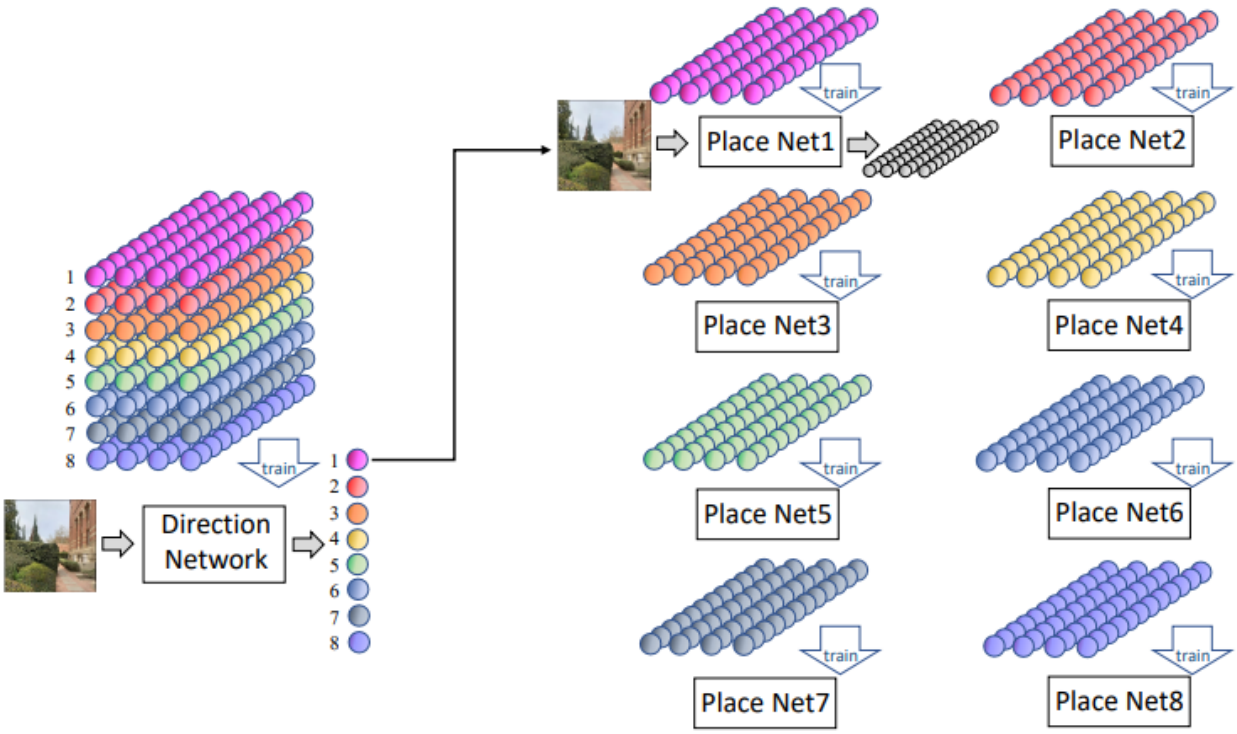
**2 Stage Process**

- All networks separate copies of the first 6 layers of AlexNet
    - Output of first has 8 categories, output of position networks have 42 outputs
- First network only predicts the direction
    - Trained on all the photos we took (except the test examples)
- Then, 8 different networks have all been trained to recognize the location after figuring out direction
    - Trained by the subset of images facing in each of those 8 directions

    *Learning how to recognize where you are when facing north doesn't transfer at all to know where you are when facing south

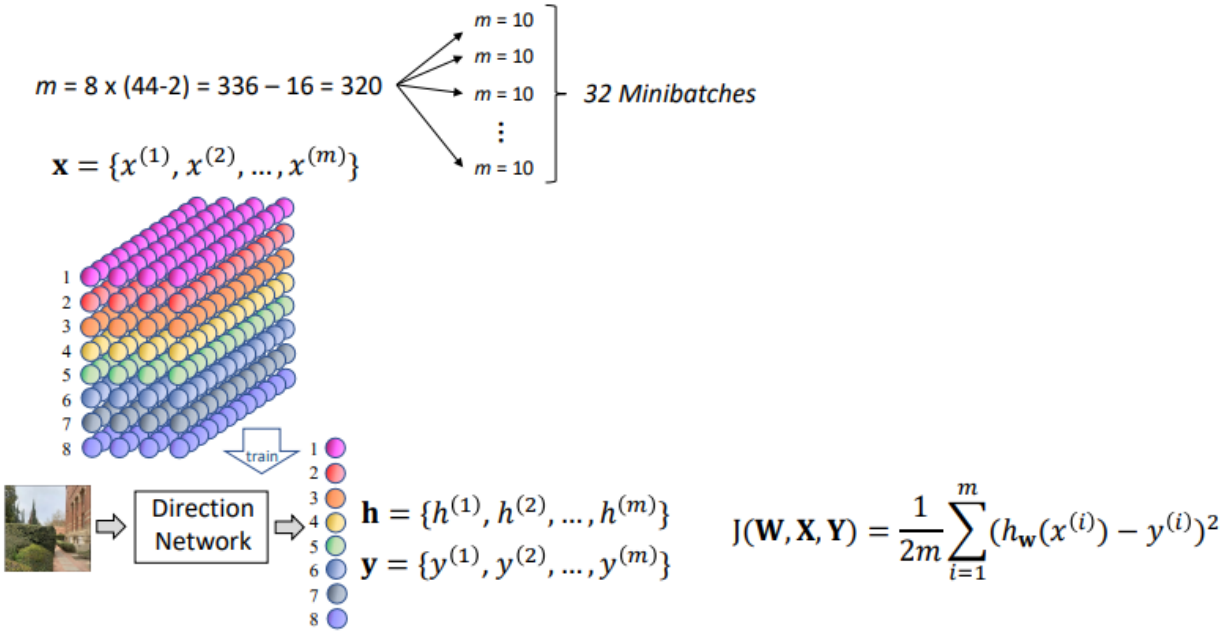



*320 images in the training set (minus testing examples)

*Two important parameters for us to optimize

- Minibatches
- Epochs

**Minibatches & Epochs**




- Cost function in many cases is the half mean squared error of the difference between the output that we want and the output that we get for each of the m training examples in the data set
- One thing these image networks do to prevent overfitting is divide the dataset up into minibatches
    - We can divide our dataset into 32 minibatches each containing 10 photos
- On each iteration of training, you randomly grab a minibatch and calculate the cost function / network error just on those (only sum from 1 to 10) and try to minimize it
- Then, grab another minibatch (w/o replacement, all new ones) and do the same
- Once you've done this for all minibatches — **1st Epoch**
- Then, you go through and do it all again with different groups/minibatches — **2nd Epoch**
    - Trained on all minibatches at a time (gradient descent after each minibatch)
    - Weights are changed after the presentation of each minibatch
    - 32 separate weight updates per Epoch in our case
    - Weights aren't reset after minibatches or epochs

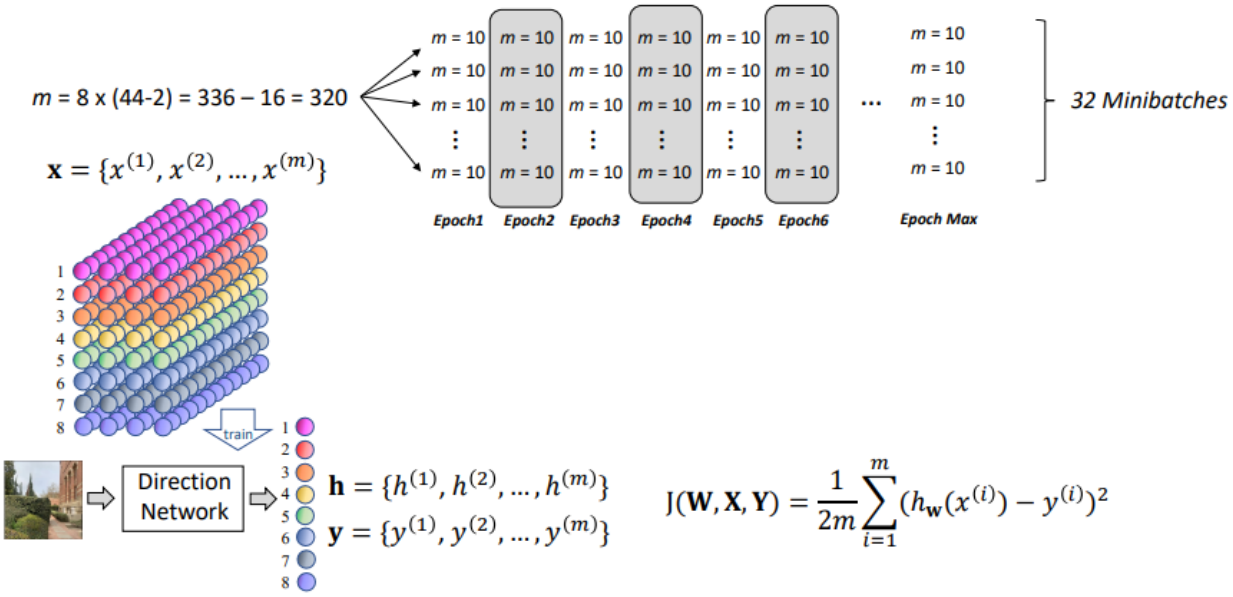
- If we're always summing over exactly the same set of training examples on every training iteration, then we can potentially fit the training dataset very perfectly (bc the sum is identical on every iteration), but may be overfit to that set of 320
- By scrambling the minibatches every epoch, slightly different cost function on every training iteration — forced to generalize to many different conditions

-

*Note:

- There would be some catastrophic interference of earlier learned minibatches by later ones within an epoch (leading to some loss of accuracy)
- But doesn't cause catastrophic interference overall because long-term, you're presenting every training example

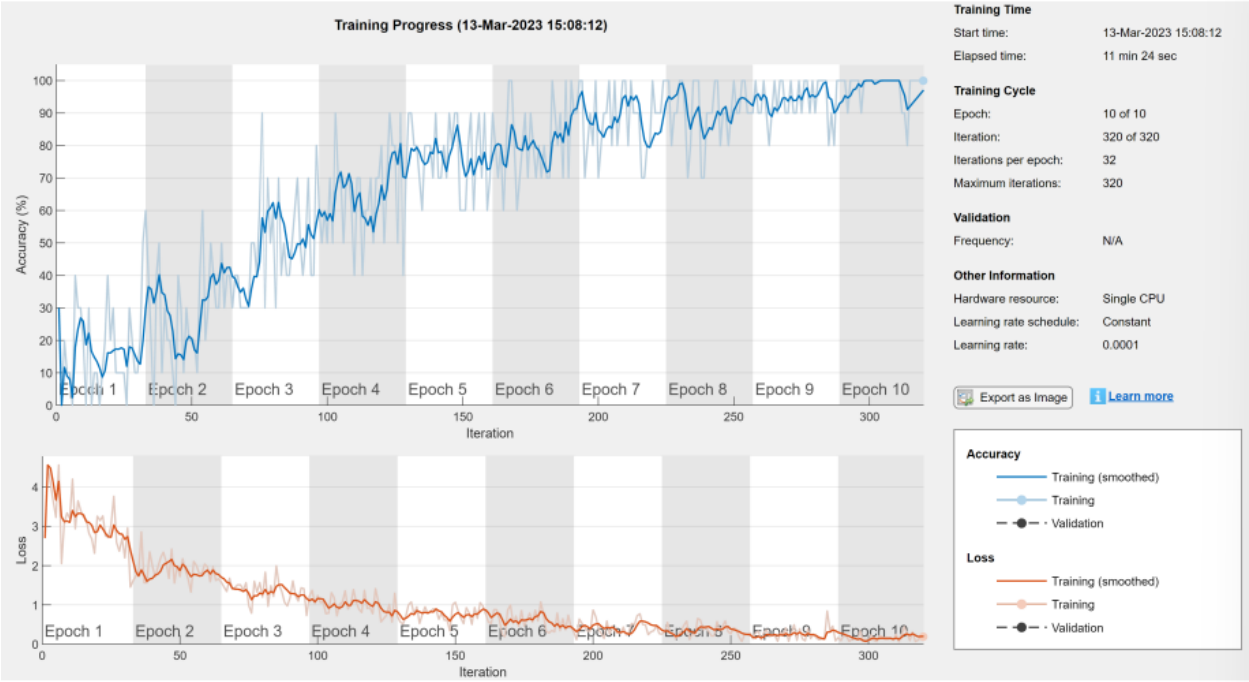
**Will have to specify how many Epochs we want to run (as well as Minibatch Size)**




- Might have to watch it and see by trial and error how many Epochs is necessary to minimize error
- Changing minibatch size will also change how long it takes to get good

*Stopping it when it gets good and not letting it keep going might be the best strategy

## Ex.) Training Dashboard

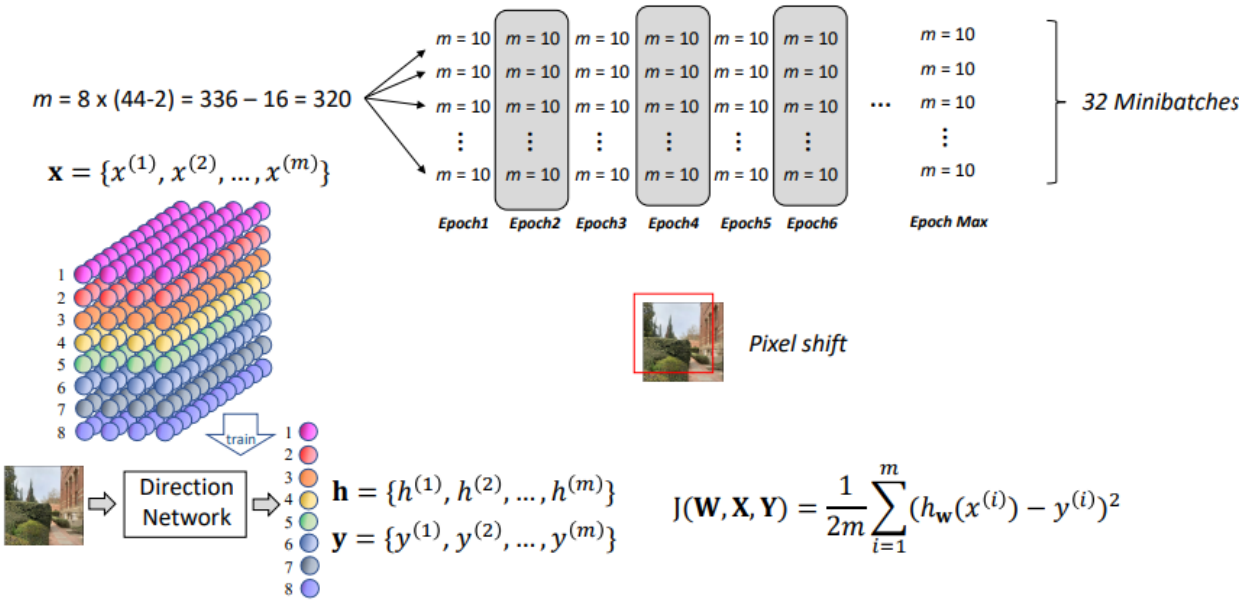



## Another Parameter: Pixel Shift

Another problem: we've undersampled our data when it comes to direction (not enough pictures/conditions)

Something we can do to expand our dataset:

- Every time we train, take the pictures and move them around a little bit
- Never quite the same picture that the network is trained on
- Helps the network to ignore what doesn't matter and generalize on what does


m = 8 x (44-2) = 336 – 16 = 320
$\mathbf{x} = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$
m = 10
Epoch1
Epoch2
Epoch3
Epoch4
Epoch5
Epoch6
Epoch Max
32 Minibatches
Pixel shift
train
Direction Network
$\mathbf{h} = \{h^{(1)}, h^{(2)}, \ldots, h^{(m)}\}$
$\mathbf{y} = \{y^{(1)}, y^{(2)}, \ldots, y^{(m)}\}$
$J(\mathbf{W}, \mathbf{X}, \mathbf{Y}) = \frac{1}{2m} \sum_{i=1}^{m} (h_{\mathbf{w}}(x^{(i)}) - y^{(i)})^2$