Now that you know the fundamentals of back prop…

Advanced Back Propagation

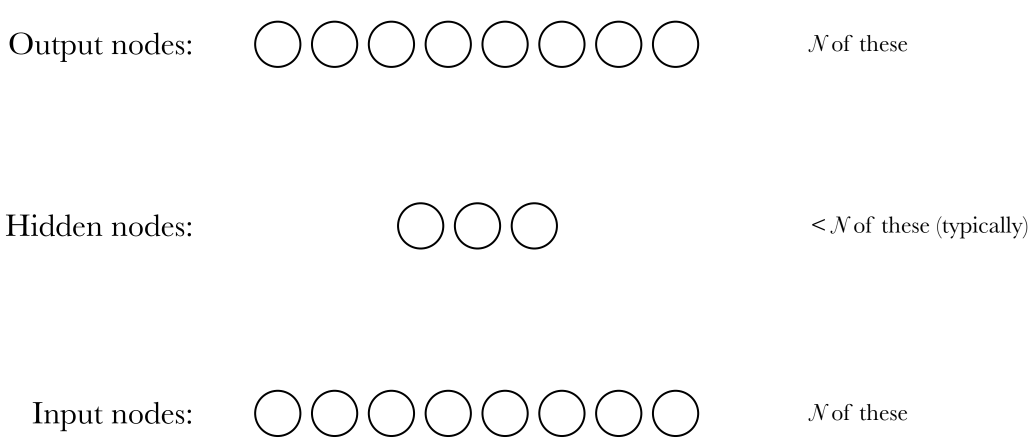
“Deep” Neural Nets

“Convolutional” Neural Nets

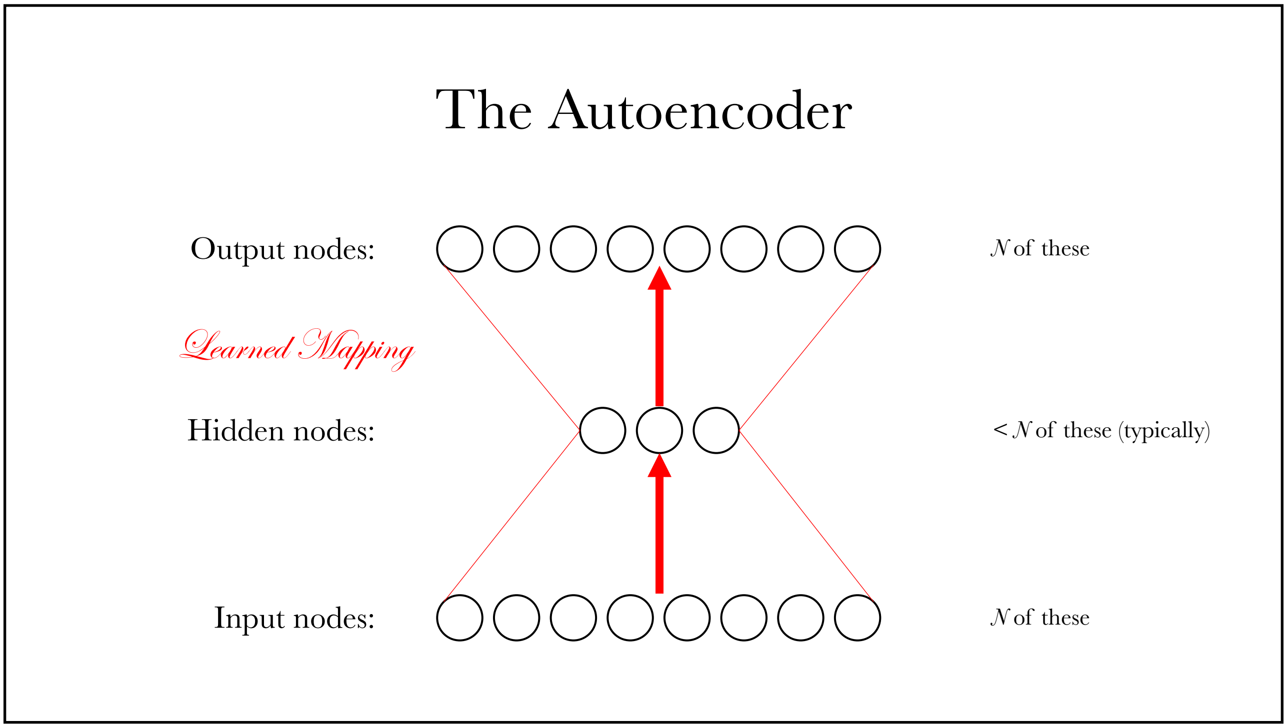
Recurrent Neural Nets

1

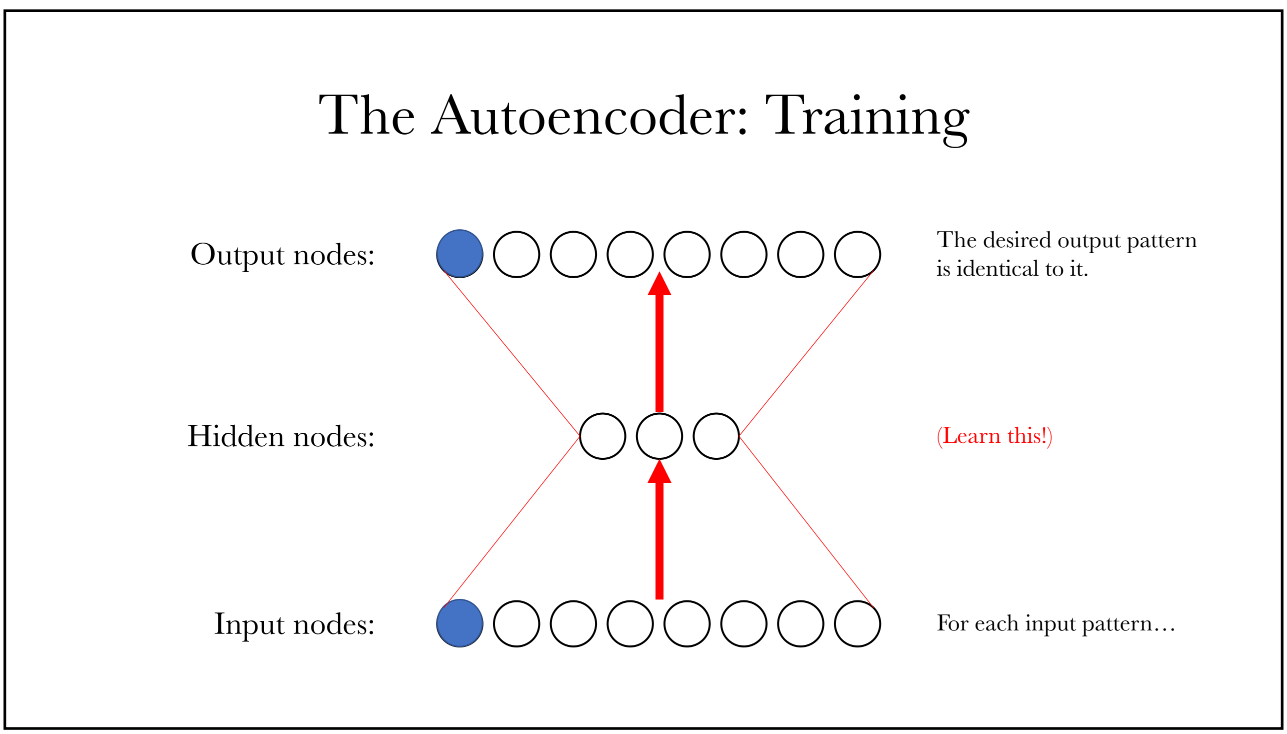
The Autoencoder



2

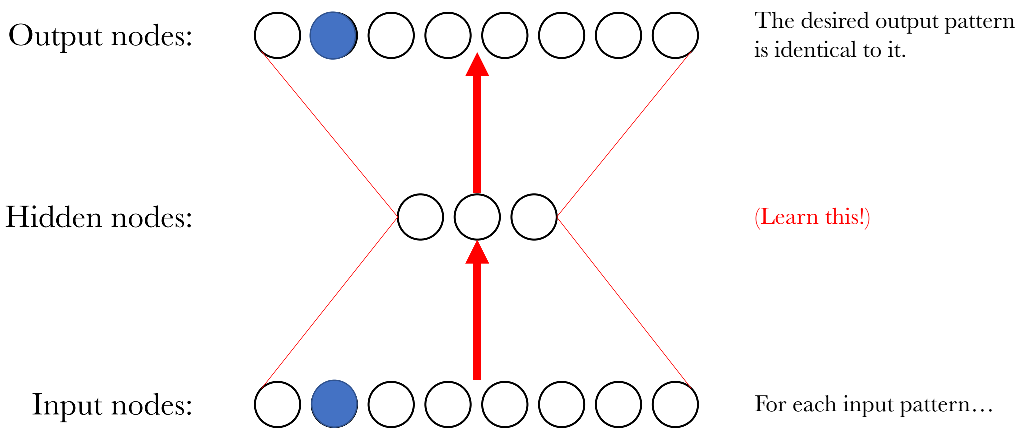

The Autoencoder
Output nodes:
N of these
Learned Mapping
Hidden nodes:
< N of these (typically)
Input nodes:
N of these

3


The Autoencoder: Training
Output nodes:
The desired output pattern is identical to it.
Hidden nodes:
(Learn this!)
Input nodes:
For each input pattern…

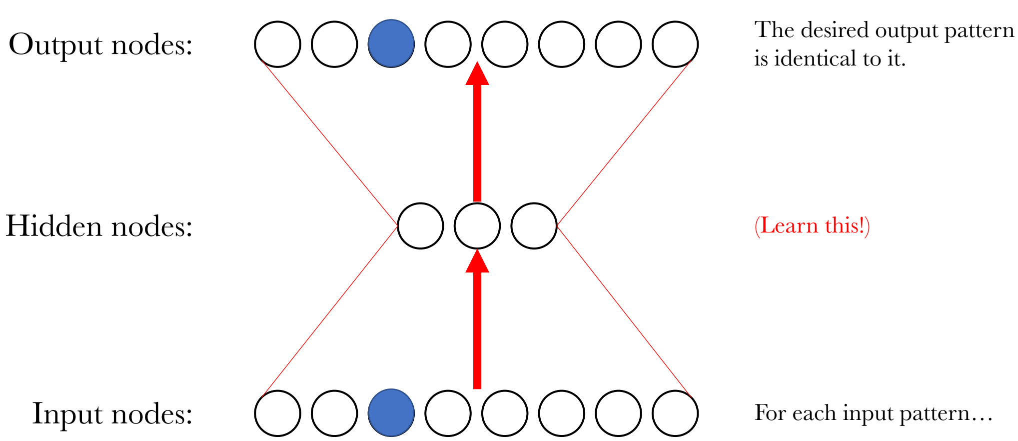
4

The Autoencoder: Training



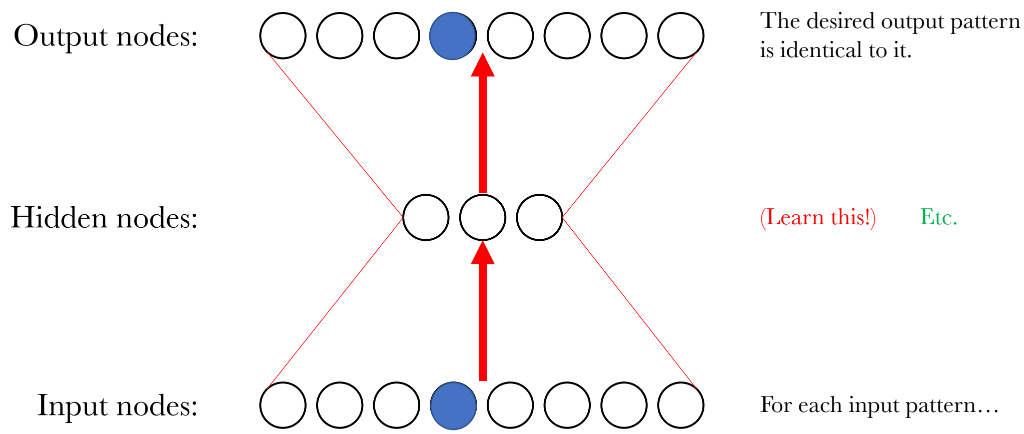
5

The Autoencoder: Training



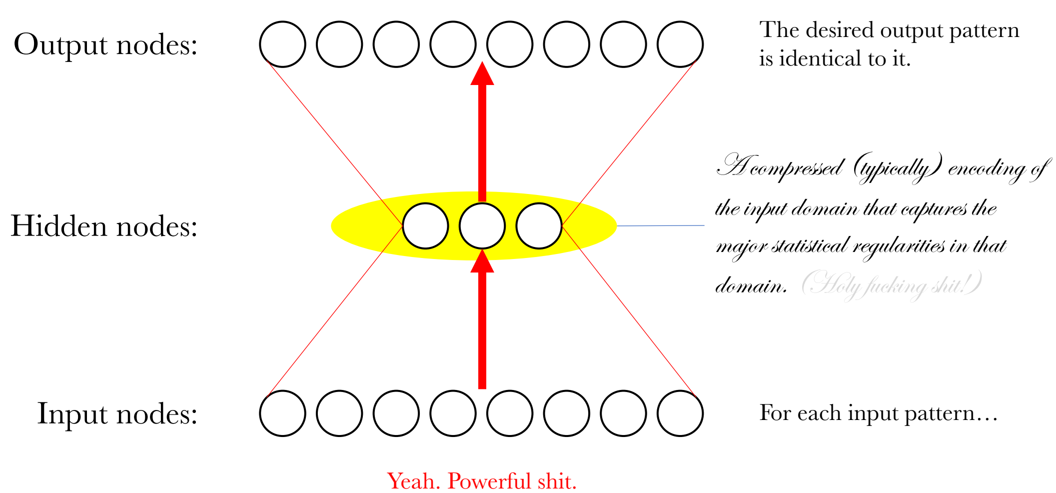
6

The Autoencoder: Training



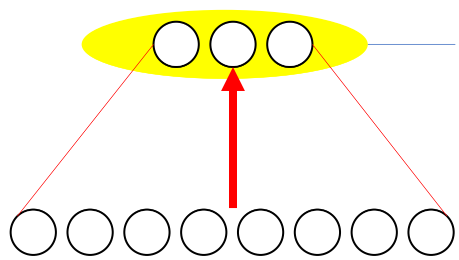
7

The Autoencoder: Result



8

The Point: An Autoencoder discovers statistical regularities in its inputs and encodes them in its "hidden" layer…



Statistical regularities.
That is, *Important Shit*

You're thinking: "Gee… what if we could use this as in input to something else?…"

Just hold your horses there, cowboy.

Now, let's change gears for a moment…

9

What's Wrong With Back Prop?

(a gazillion things. But if you're an engineer…)

- The #$&^*@! *derivative*: $\Delta w_{ij} = f(error * a_i * (1 - a_i))$
  - *Goes to zero as activation approaches zero or one.*
- Happens in each layer *successively*
- Result: *Errors get washed out to near nothing after just a few layers.*
  - ➢ *Many-layered perceptrons therefore take forever to train.*
- But some problems are solved more easily with more layers ☹
  - What is a poor ~~hack~~ computer programmer to do??
  - This problem made back prop ~~unprofitable~~ unpopular for several years. *Until*…)

10

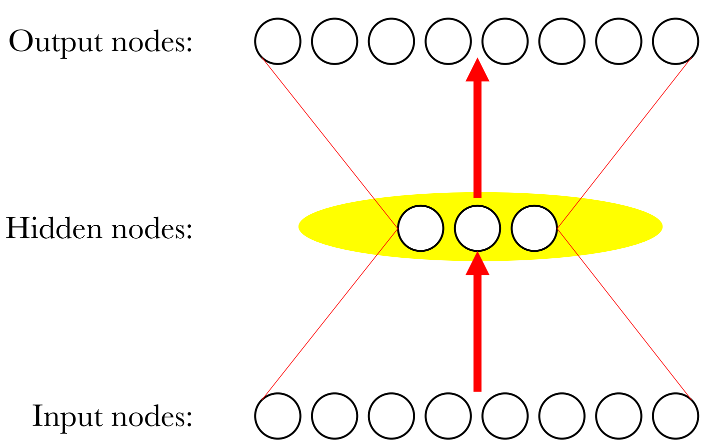
An Idea!

- Geoff Hinton: *What if we pre-train a neural net with an auto-encoder, and then use* ***that*** *as the input layer to another network?*
  - (This is the idea you had earlier)
- *And what if we do this over and over?!*
- *Like, with lots of hidden nodes and lots of hidden layers?*
- *Kinda like…*

11

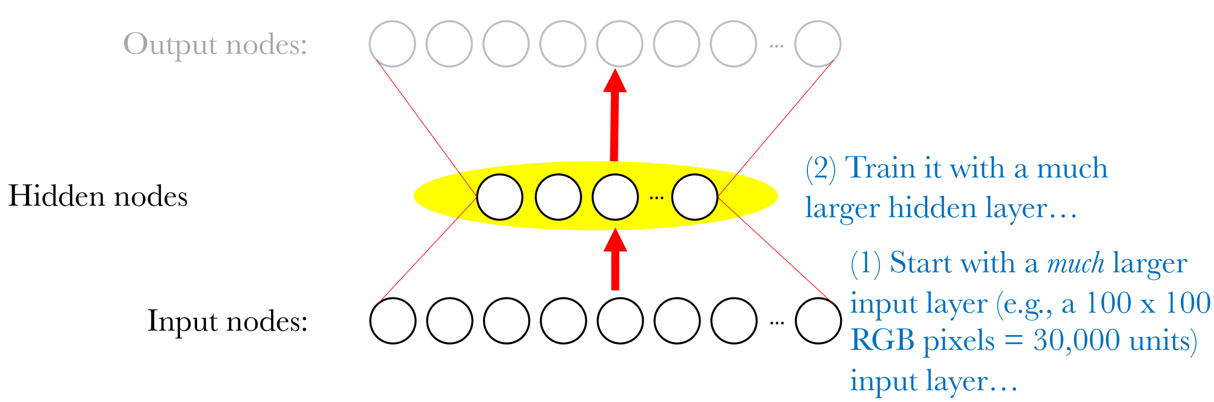
An Idea!

Start with an autoencoder, *but…*



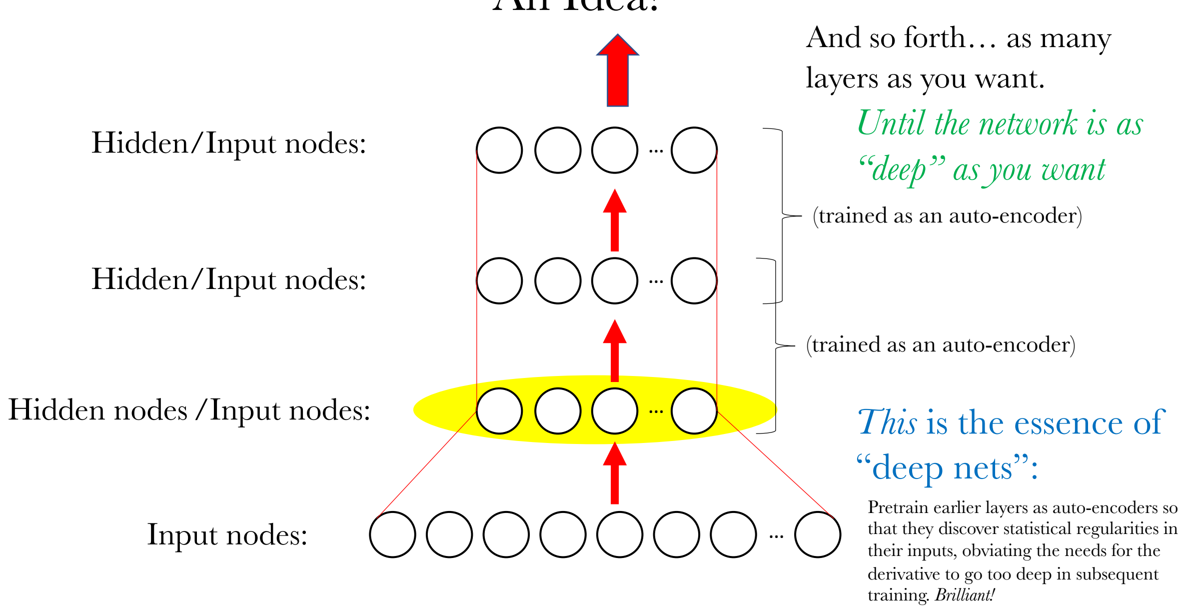
12

An Idea!



13

An Idea!



14

"Convolutional" Nets

It's just weight duplication (aka "weight sharing", "error sharing", "update sharing").

Start with an image.

Take, say, a 10 X 10 pixel piece of the image…

and train an auto-encoder on it.

15

At each 10 X 10 Window Over the Image…

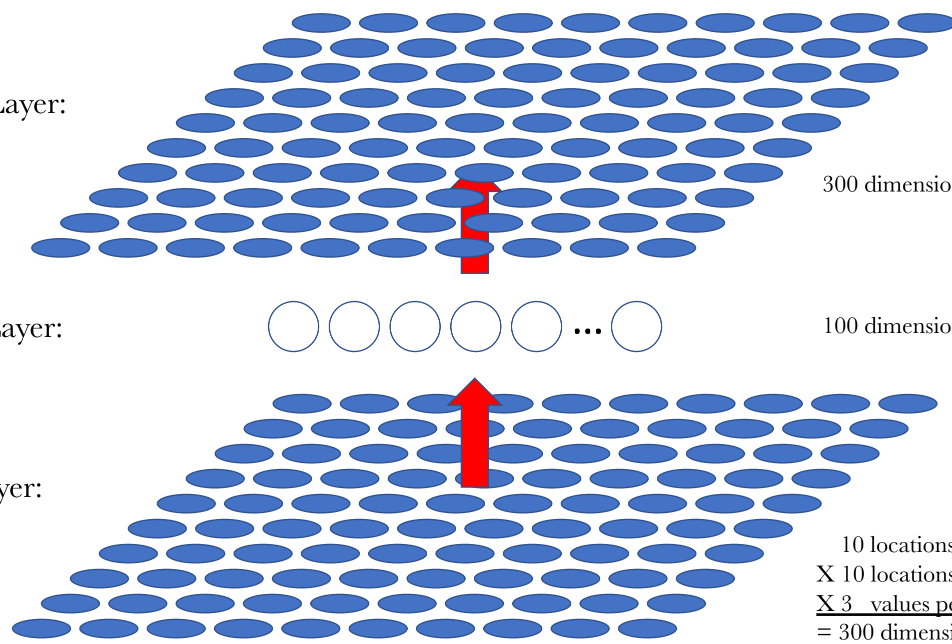
Output Layer:

300 dimensions

Hidden Layer:

100 dimensions (say)

Input Layer:



10 locations
X 10 locations
X 3 values per location
= 300 dimensions

16

"Convolutional" Nets

It's just weight duplication (aka "weight sharing", "error sharing", "update sharing").

Start with an image.

Take, say, a 10 X 10 pixel piece of the image…

and train an auto-encoder on it.

Move it over 5 pixels and repeat. Over and over. Until you've covered the whole image.

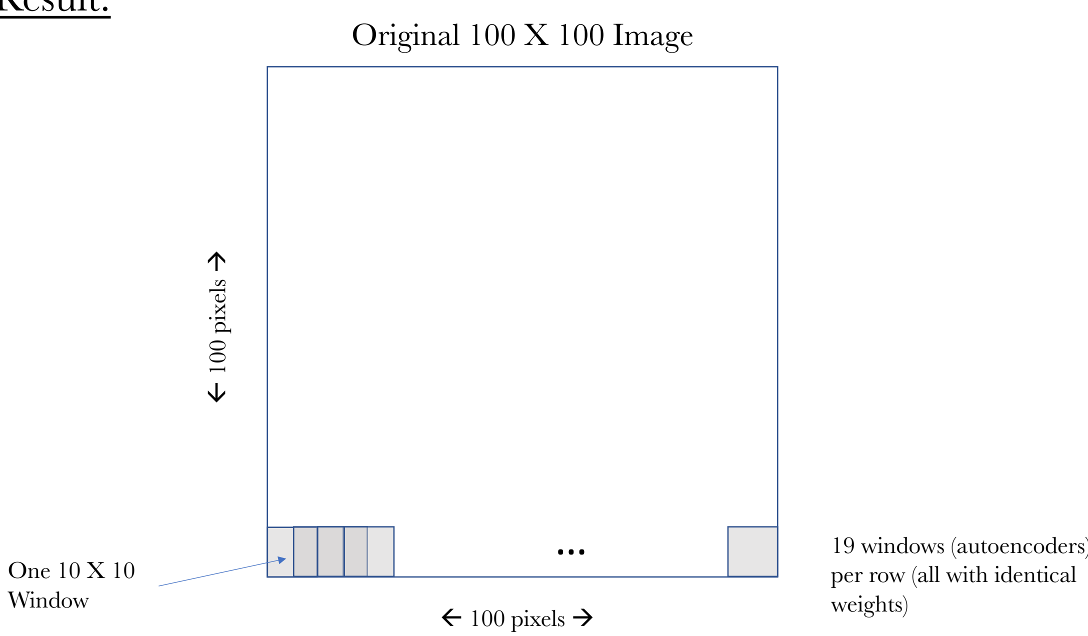
Repeat for multiple images.

The trick: *Weight sharing:* It's the same weight matrix over and over. One weight matrix gets to learn from every location.

Result (sometimes): The weight matrix learns "V1-like" RFs.

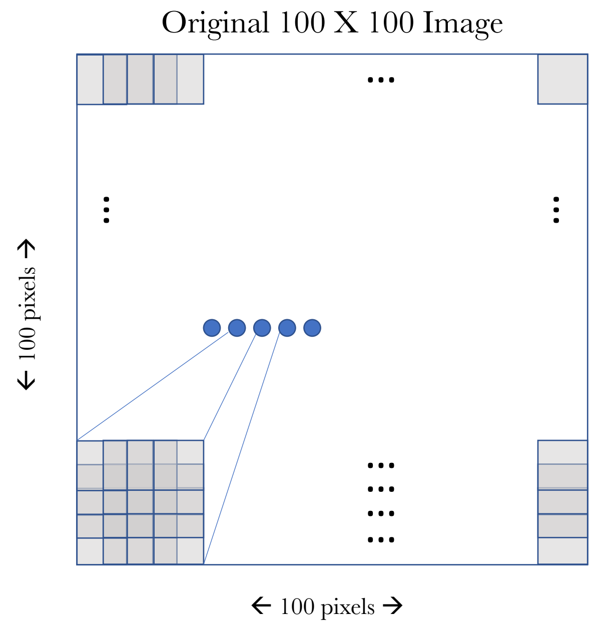
17

The Result:



18

The Result:



19 Rows
X 19 windows per row
X 100 units per window
= 36,100 units

(compare to 100 X 100 X 3 = 30,000 pixel values in the original image)

36,100 units to represent the major statistical regularities in 10 X 10 windows over the image

19 windows (autoencoders) per row (all with identical weights)

19

"Convolutional" Nets

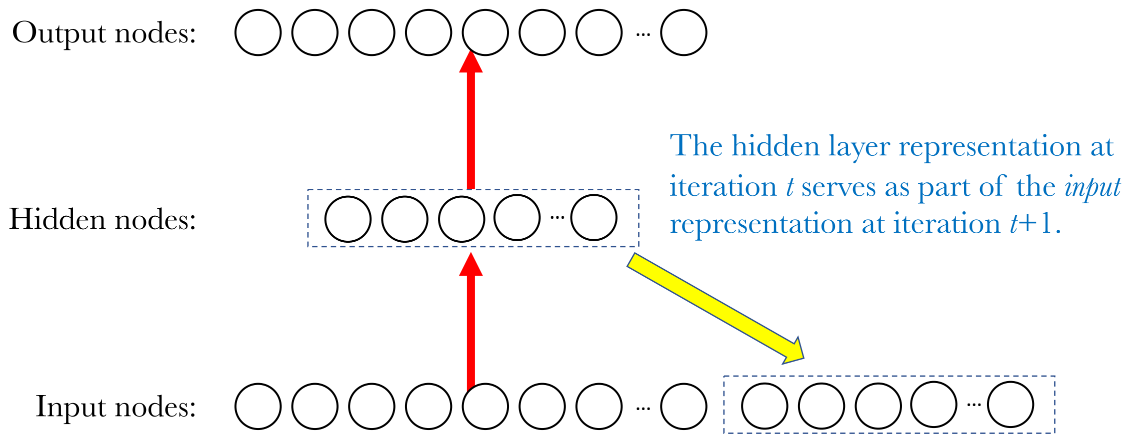
Armed with that recoding of the image (an *N* X *N* matrix of *M* "V1-like" units)…

- Use *that* as the input to a "deep" net…
- Or do "pooling" over adjacent locations (*not* Poggio's idea; see Fukushima & Miyaki, 1982)…
- Or train each location with an autoencoder…
- Or train multiple adjacent locations with an autoencoder…
- Or do… whatever your little capitalist heart desires.
- *Much* money to be made here, but no intellectual progress. (An exercise for dupes. Preferably dupes with money.)

20

Recurrent Nets

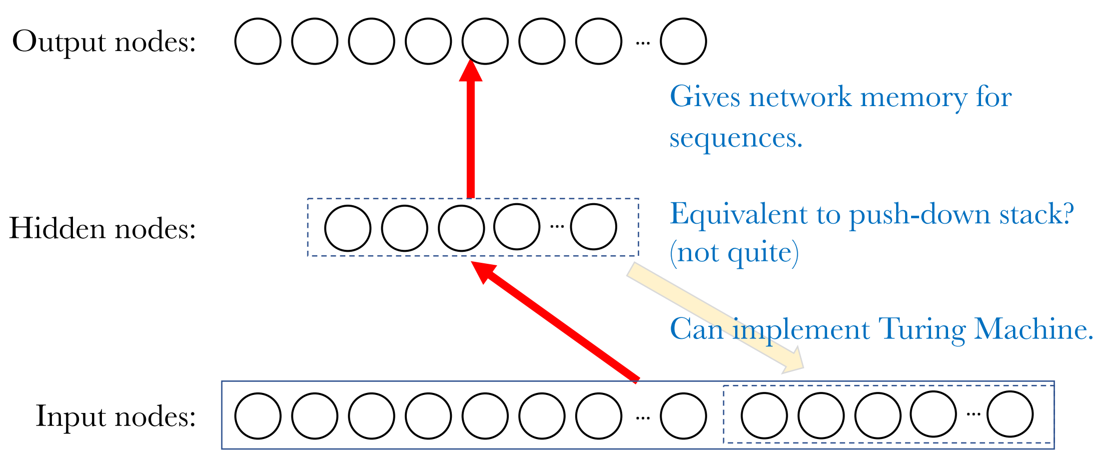
An ordinary back prop net, *but…*



21

Recurrent Nets

An ordinary back prop net, *but…*



22