People telling me AI is going to destroy the world

My neural network

Organizational

- Homework: **You nearly did it!**
- Homework 7 to be released next Wednesday (you will have until Dec 6th)

→ No more extensions

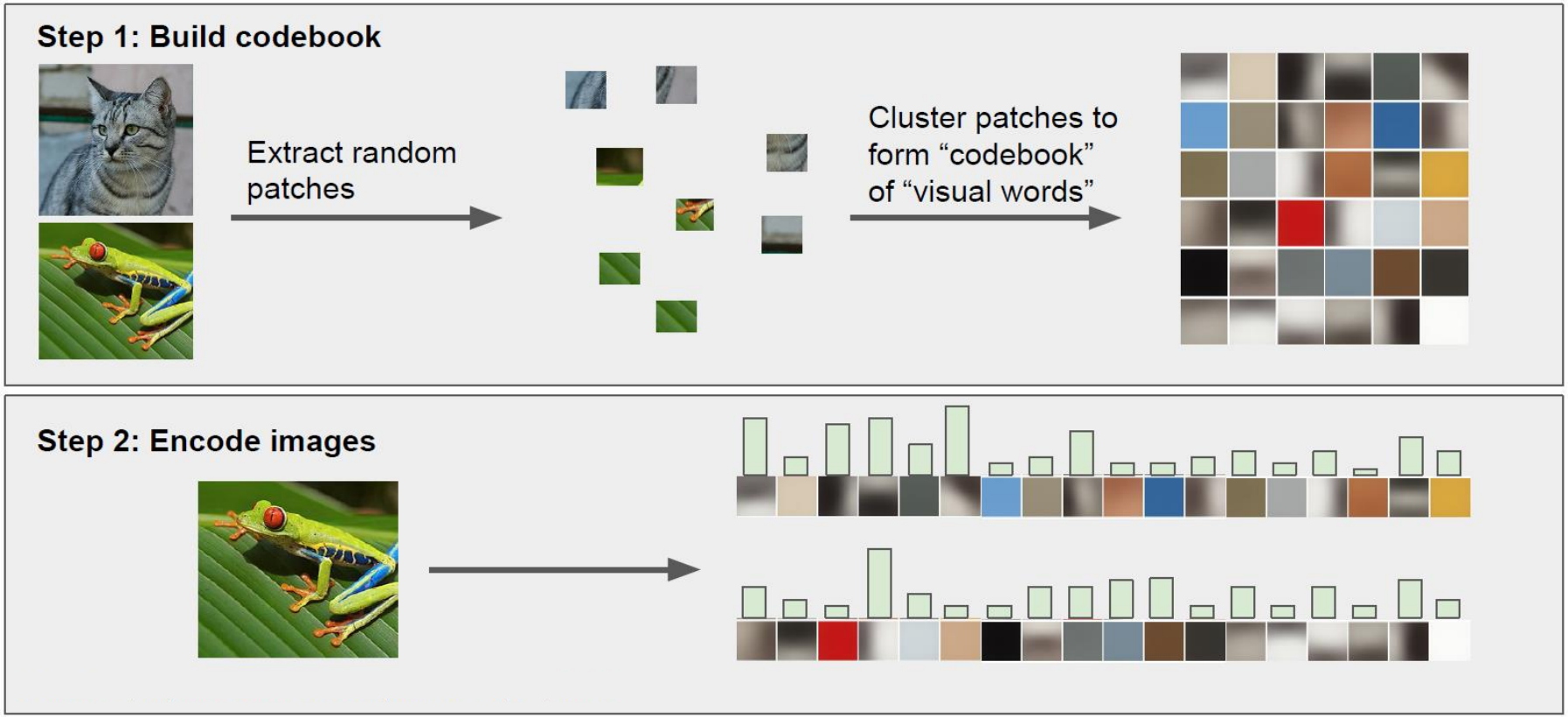
Midterm Wednesday 11/29 4.30 to 5.45

- Instructions were released
- Proposals due Nov 8th
- Get to work!
  - Check boxes for grading first
  - Then, try to have a little fun =)

Interpretability, Generalization and Domain Gaps

Bag of Features - BagNet

Bag-of-Features?



Fei-Fei, L., & Perona, P. (2005) A bayesian hierarchical model for learning natural scene categories. CVPR

Images from Stanford cs231n.

Bag-of-Features?

Deep Features

- Input Q*Q px image patch
- BagNet generates 2048-dim feature vector
- Linear classifier on this 2048-dim vector
- After linear classification: #classes heatmaps of “class evidence”
- Average class evidence and pass into softmax → Probability

Brendel, W., & Bethge, M. (2018). Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet. ICLR.

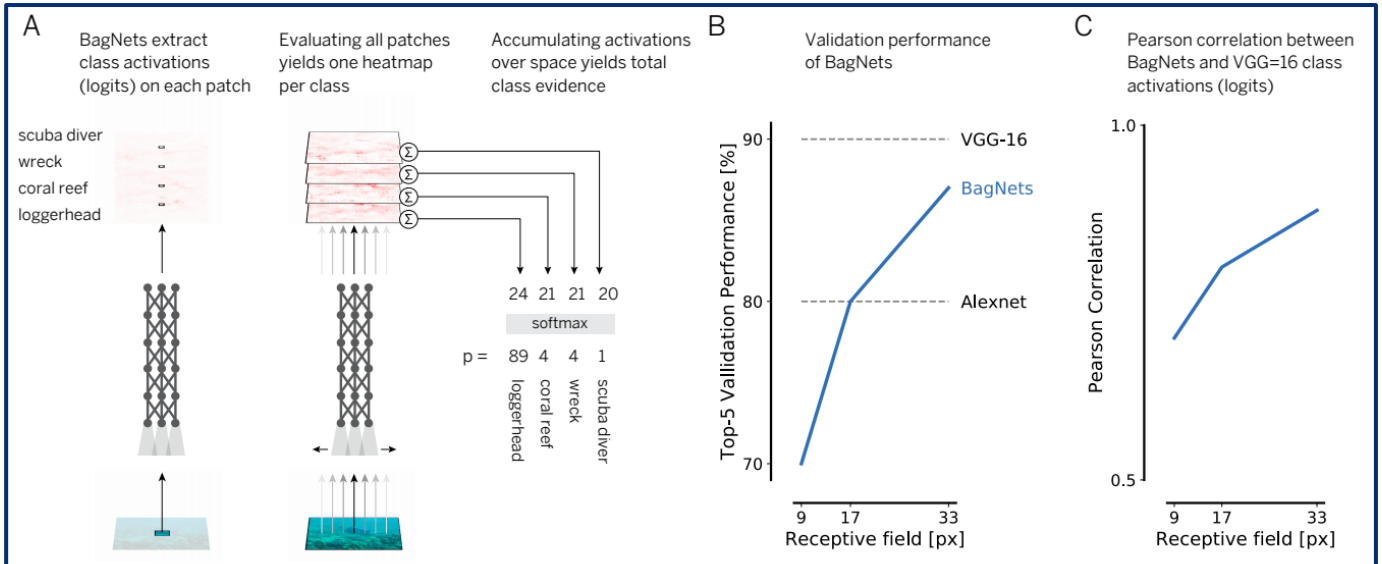
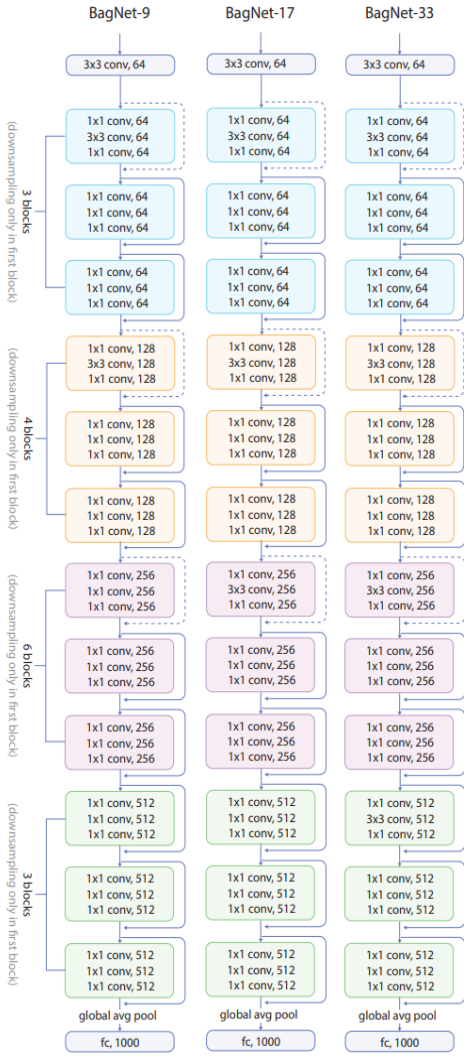
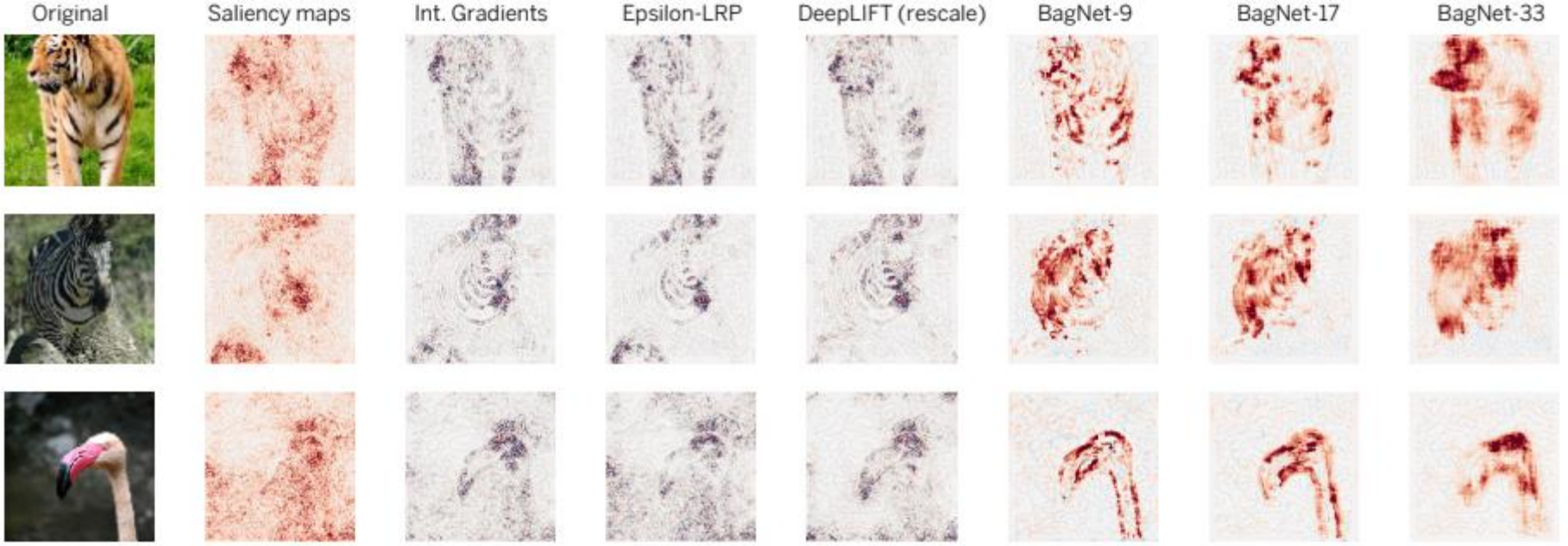


Figure 1: Deep bag-of-features models (BagNets). (A) The models extract features from small image patches which are each fed into a linear classifier yielding one logit heatmap per class. These heatmaps are averaged across space and passed through a softmax to get the final class probabilities. (B) Top-5 ImageNet performance over patch size. (C) Correlation with logits of VGG-16.



- BagNet architecture is ResNet like with 1x1 conv to limit receptive field
- Surprisingly good performance!

Brendel, W., & Bethge, M. (2018). Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet. ICLR.



- Similarities in “decision making” (saliency) to SOTA models suggest that current network architectures base their decisions on a large number of weak and local statistical regularities

- One way forward is to define novel tasks that cannot be solved using local statistical regularities

Brendel, W., & Bethge, M. (2018). Approximating CNNs with Bag-of-local-Features models works surprisingly well on ImageNet. ICLR.

Interpretability, Generalization and Domain Gaps

Deep Image Prior

Deep Image Prior

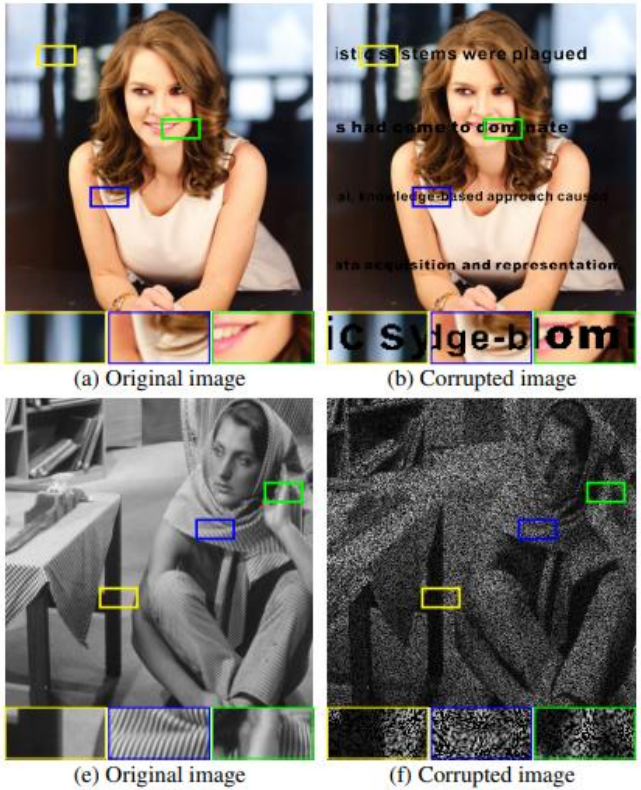
- Image restoration $x^\star = \min_x E(x, x_0) + R(x)$, where

  E is a task-dependent data term (similarity),
  x_0 is a corrupted observation (noise, occlusion, etc), and
  R is a regularizer (e.g. total variation, L2, wavelets,…)

- Regularizer is replaced by a CNN

$$\theta^\star = \min_\theta E(f_\theta(z), x_0), \quad x^\star = f_{\theta^\star}(z)$$

  Minimizer is optimized using a random z, starting from random initialization of θ



Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2018). Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 9446-9454).

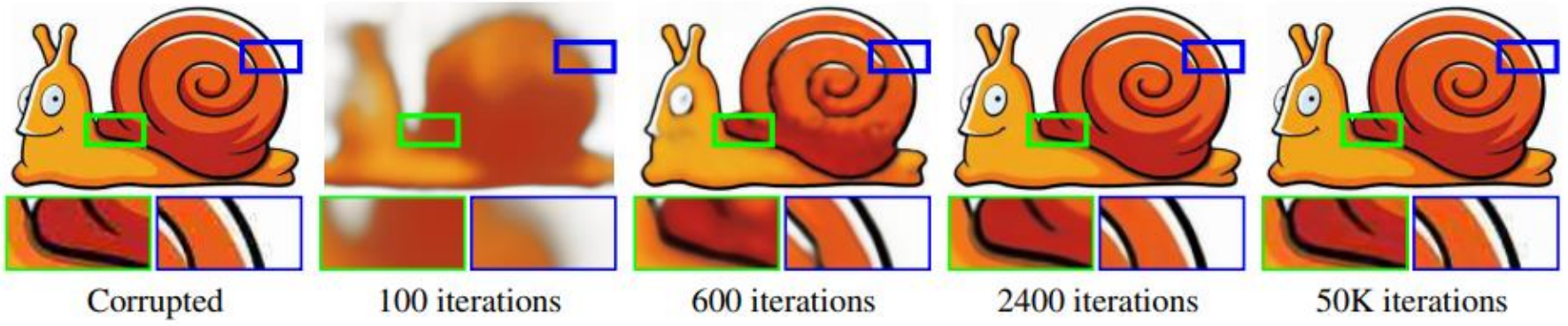


Figure 3: **Blind restoration of a JPEG-compressed image.** (*electronic zoom-in recommended*) Our approach can restore an image with a complex degradation (JPEG compression in this case). As the optimization process progresses, the deep image prior allows to recover most of the signal while getting rid of halos and blockiness (after 2400 iterations) before eventually overfitting to the input (at 50K iterations).

- Optimization is fast for “natural images” without corruption
- CNN can fit corruption, but very reluctantly

- Parametrization (CNN architecture) has high impedance to noise and low impedance to signal!

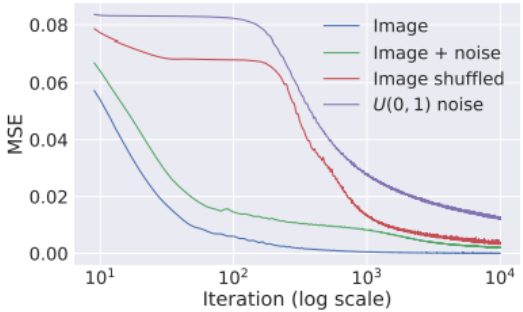
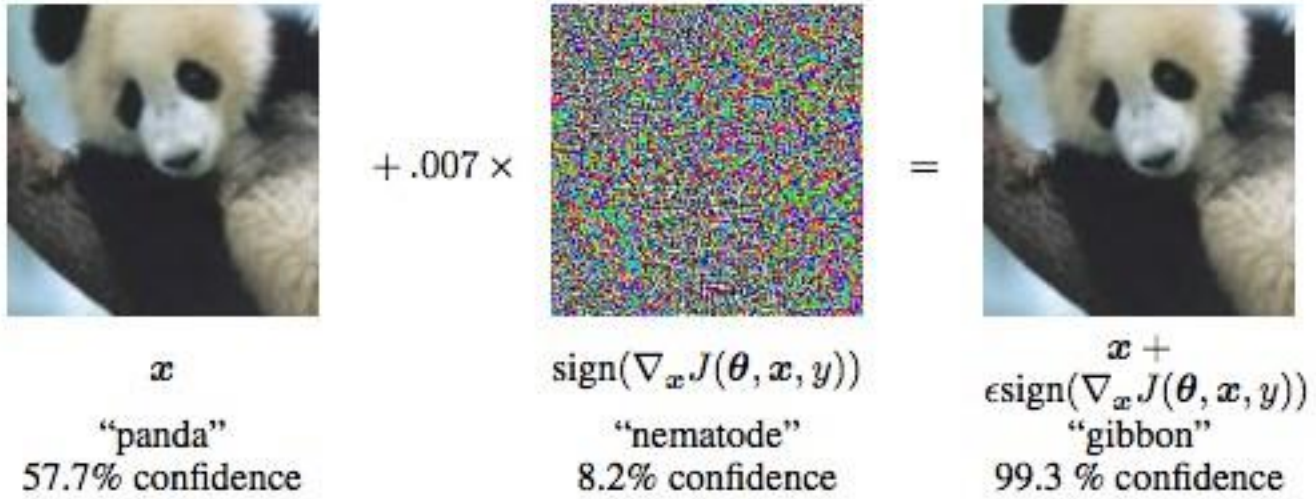


Figure 2: Learning curves for the reconstruction task using: a natural image, the same plus i.i.d. noise, the same randomly scrambled, and white noise. Naturally-looking images result in much faster convergence, whereas noise is rejected.

Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2018). Deep image prior. CVPR (pp. 9446-9454).

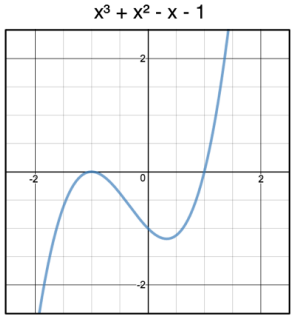

MY NEURAL NET
DURING
ADVERSARIAL ATTACKS
imgflip.com

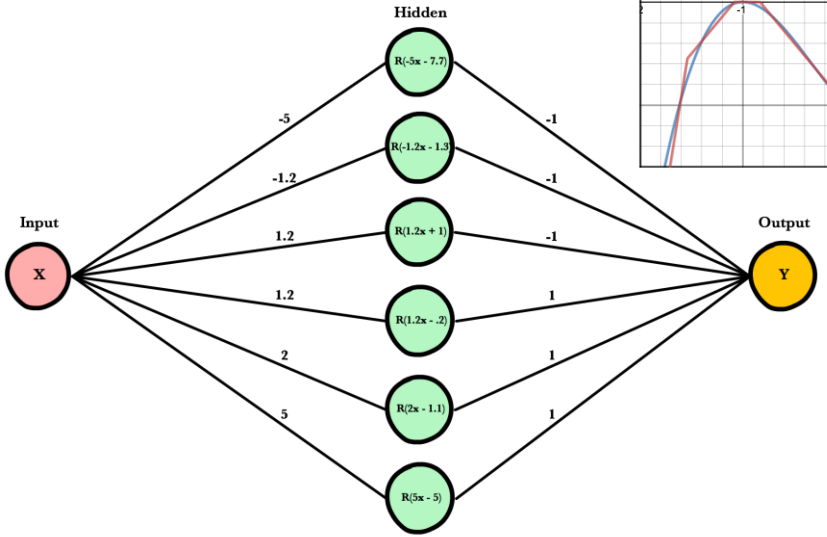
Adversarial Examples: What are they?



Understanding Universal Approximation

- f(W,x): ReLU activation; W in R(3x1), b in R(3x1), and x in R(1x1)
  → 3 functions n_i (shifted ReLUs)
- Output neuron: Combine the 3 functions





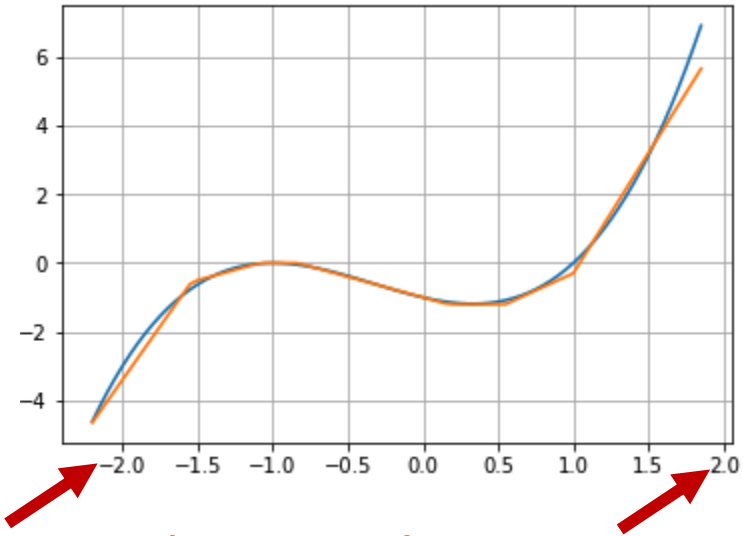
$$n_1(x) = Relu(-5x - 7.7)$$
$$n_2(x) = Relu(-1.2x - 1.3)$$
$$n_3(x) = Relu(1.2x + 1)$$
$$n_4(x) = Relu(1.2x - .2)$$
$$n_5(x) = Relu(2x - 1.1)$$
$$n_6(x) = Relu(5x - 5)$$

$$Z(x) = -n_1(x) - n_2(x) - n_3(x) + n_4(x) + n_5(x) + n_6(x)$$

Example taken from a blog post: link.

Compact Domains

Another example



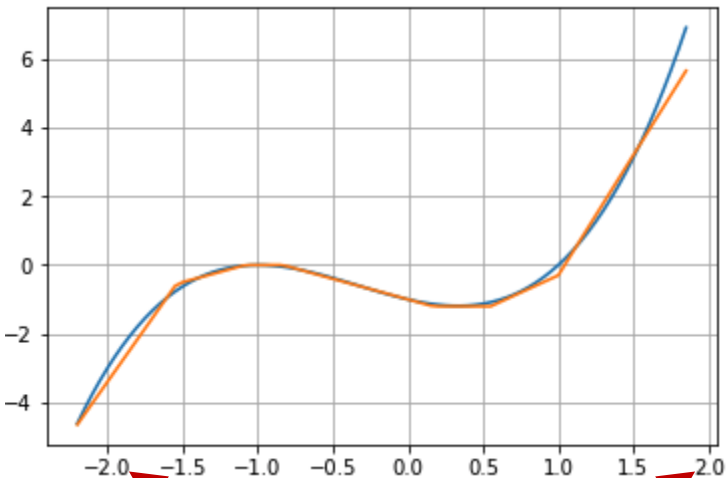
Note the range of x!

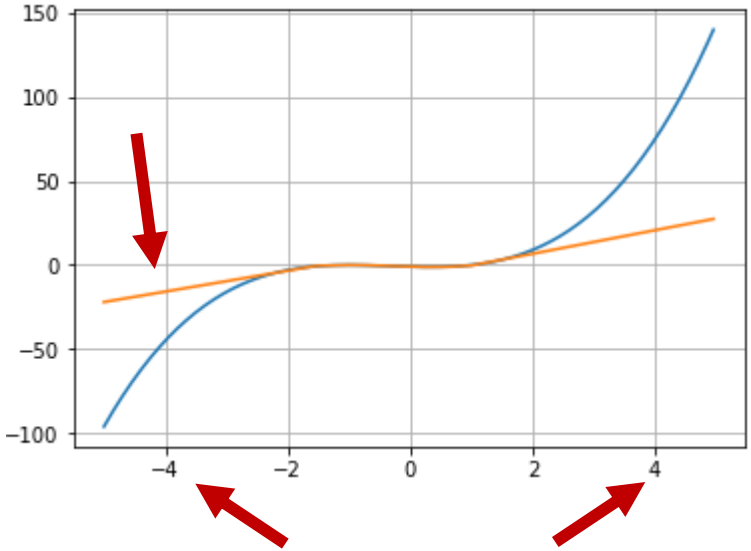
```
def feed_forward(input):
    Zh = np.dot(input, Wh)
    Ah = relu(Zh + Bh)
    Zo = np.dot(Ah, Wo)
    Ao = Zo + Bo
    return Ao
```

Example taken from a blog post: link.

Compact Domains

Another example





Note the range of x!

What happened?

Example taken from a blog post: link.