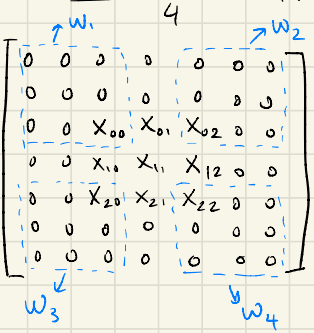
1. $W = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ w_{10} & w_{11} & w_{12} \\ w_{20} & w_{21} & w_{22} \end{bmatrix}$ $\quad X = \begin{bmatrix} x_{00} & x_{01} & x_{02} \\ x_{10} & x_{11} & x_{12} \\ x_{20} & x_{21} & x_{22} \end{bmatrix}$

Stride = 4 $\quad$ padding = 2

output size = $\frac{3-3+2(2)}{4} + 1 = 2 \Rightarrow$ 2x2 matrix

$X_{padded} =$



$w_1 = w_{22} x_{00}$ $\quad w_2 = w_{20} x_{02}$

$w_3 = w_{02} x_{20}$ $\quad w_4 = w_{00} x_{22}$

$wx = \begin{bmatrix} w_{22}x_{00} & w_{20}x_{02} \\ w_{02}x_{20} & w_{00}x_{22} \end{bmatrix}$

$Y = [w_{22}x_{00} \quad w_{20}x_{02} \quad w_{02}x_{20} \quad w_{00}x_{22}]^T$

$Y = \underbrace{\begin{bmatrix} w_{22} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{20} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_{02} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & w_{00} \end{bmatrix}}_{A} \begin{bmatrix} x_{00} \\ x_{01} \\ x_{02} \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{20} \\ x_{21} \\ x_{22} \end{bmatrix} = Ax$

2. $W = \begin{bmatrix} w_{00} & w_{01} \\ w_{10} & w_{11} \end{bmatrix}$ $X = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix}$

stride = 2

define w as

$W = \begin{bmatrix} w_{00} & w_{01} & w_{00} & w_{01} \\ w_{10} & w_{11} & w_{10} & w_{11} \\ w_{00} & w_{01} & w_{00} & w_{01} \\ w_{10} & w_{11} & w_{10} & w_{11} \end{bmatrix} = \begin{bmatrix} x_{00}w_{00} & x_{00}w_{01} & x_{01}w_{00} & x_{01}w_{01} \\ x_{00}w_{10} & x_{00}w_{11} & x_{01}w_{10} & x_{01}w_{11} \\ x_{10}w_{00} & x_{10}w_{01} & x_{11}w_{00} & x_{11}w_{01} \\ x_{10}w_{10} & x_{10}w_{11} & x_{11}w_{10} & x_{11}w_{11} \end{bmatrix}$

row-wise $w = [x_{00}w_{00}\ x_{00}w_{01}\ x_{01}w_{00}\ x_{01}w_{01}\ x_{00}w_{10}\ x_{00}w_{11}\ x_{01}w_{10}\ x_{01}w_{11}\ \ldots\ x_{11}w_{11}]$

$Y = \begin{bmatrix} w_{00} & 0 & 0 & 0 \\ w_{01} & 0 & 0 & 0 \\ 0 & w_{00} & 0 & 0 \\ 0 & w_{01} & 0 & 0 \\ w_{10} & 0 & 0 & 0 \\ w_{11} & 0 & 0 & 0 \\ 0 & w_{10} & 0 & 0 \\ 0 & w_{11} & 0 & 0 \\ 0 & 0 & w_{00} & 0 \\ 0 & 0 & w_{01} & 0 \\ 0 & 0 & 0 & w_{00} \\ 0 & 0 & 0 & w_{01} \\ 0 & 0 & w_{10} & 0 \\ 0 & 0 & w_{11} & 0 \\ 0 & 0 & 0 & w_{10} \\ 0 & 0 & 0 & w_{11} \end{bmatrix} \begin{bmatrix} x_{00} \\ x_{01} \\ x_{10} \\ x_{11} \end{bmatrix} = Ax$

A

3. Kernel $(o \times r^2, i, k, k) \equiv$ Kernel $(o, i, k \times r, k \times r)$

$o = 1 \quad r = 2 \quad k = 1$

$\Rightarrow \quad (4, 1, 1, 1) \equiv (1, 1, 2, 2)$

For w of size $(4,1,1,1)$

$$w \Rightarrow \quad w_1 = [w_1] \quad w_2 = [w_2] \quad w_3 = [w_3] \quad w_4 = [w_4] \qquad X = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix}$$

$$Y = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ 0 & w_1 & 0 & 0 \\ 0 & 0 & w_1 & 0 \\ 0 & 0 & 0 & w_1 \\ w_2 & 0 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ 0 & 0 & w_2 & 0 \\ 0 & 0 & 0 & w_2 \\ w_3 & 0 & 0 & 0 \\ 0 & w_3 & 0 & 0 \\ 0 & 0 & w_3 & 0 \\ 0 & 0 & 0 & w_3 \\ w_4 & 0 & 0 & 0 \\ 0 & w_4 & 0 & 0 \\ 0 & 0 & w_4 & 0 \\ 0 & 0 & 0 & w_4 \end{bmatrix} \begin{bmatrix} x_{00} \\ x_{01} \\ x_{10} \\ x_{11} \end{bmatrix} = \begin{bmatrix} w_1 x_{00} \\ w_1 x_{01} \\ w_1 x_{10} \\ w_1 x_{11} \\ w_2 x_{00} \\ w_2 x_{01} \\ w_2 x_{10} \\ w_2 x_{11} \\ w_3 x_{00} \\ w_3 x_{01} \\ w_3 x_{10} \\ w_3 x_{11} \\ w_4 x_{00} \\ w_4 x_{01} \\ w_4 x_{10} \\ w_4 x_{11} \end{bmatrix}$$

For w of size $(1, 1, 2, 2)$

$$w = \begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix} \qquad X = \begin{bmatrix} x_{00} & x_{01} \\ x_{10} & x_{11} \end{bmatrix}$$

From q^2 we can write:

$$Y = \begin{bmatrix} w_1 & 0 & 0 & 0 \\ w_2 & 0 & 0 & 0 \\ 0 & w_1 & 0 & 0 \\ 0 & w_2 & 0 & 0 \\ w_3 & 0 & 0 & 0 \\ w_4 & 0 & 0 & 0 \\ 0 & w_3 & 0 & 0 \\ 0 & w_4 & 0 & 0 \\ 0 & 0 & w_1 & 0 \\ 0 & 0 & w_2 & 0 \\ 0 & 0 & 0 & w_1 \\ 0 & 0 & 0 & w_2 \\ 0 & 0 & w_3 & 0 \\ 0 & 0 & w_4 & 0 \\ 0 & 0 & 0 & w_3 \\ 0 & 0 & 0 & w_4 \end{bmatrix} \begin{bmatrix} x_{00} \\ x_{01} \\ x_{10} \\ x_{11} \end{bmatrix} = \begin{bmatrix} w_1 x_{00} \\ w_2 x_{00} \\ w_1 x_{01} \\ w_2 x_{01} \\ w_3 x_{00} \\ w_4 x_{00} \\ w_3 x_{01} \\ w_4 x_{01} \\ w_1 x_{10} \\ w_2 x_{10} \\ w_1 x_{11} \\ w_2 x_{11} \\ w_3 x_{10} \\ w_4 x_{10} \\ w_3 x_{11} \\ w_4 x_{11} \end{bmatrix}$$

We can see that both Y's have same elements, just in different order

4. $f(x) = \begin{cases} 1 & \text{if } w^T x + b \geq 0 \\ 0 & \text{if } w^T x + b < 0 \end{cases}$

x_1	x_2	$f_{AND}(x)$
0	0	0
0	1	0
1	0	0
1	1	1

$w_{AND} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $b_{AND} = -2$

$f\left(w_{AND}^T \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b_{AND}\right) = f_{AND}(x)$

$x_1 = 0, x_2 = 1 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 2 = -1 \Rightarrow 0$

$x_1 = 1, x_2 = 0 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 2 = -1 \Rightarrow 0$

$x_1 = 1, x_2 = 1 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 2 = 0 \Rightarrow 1$

$x_1 = 0, x_2 = 0 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 2 = -2 \Rightarrow 0$

$w_{OR} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $b_{OR} = -0.1$

$x_1 = 0, x_2 = 1 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} - 0.1 = 0.9 \Rightarrow 1$

$x_1 = 1, x_2 = 0 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} - 0.1 = 0.9 \Rightarrow 1$

$x_1 = 1, x_2 = 1 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.1 = 1.9 \Rightarrow 1$

$x_1 = 0, x_2 = 0 \Rightarrow \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.1 = -0.1 \Rightarrow 0$

5.

x_1	x_2	$f_{XOR}(x)$
0	0	0
0	1	1
1	0	1
1	1	0

proof by contradiction:

If $f(x) = w^T x + b$ can output a linear model function

↓ Let $x_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ $x_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Then $f(x_1) < f(x_2)$

$f\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) < f\left(\begin{bmatrix} 0 \\ 1 \end{bmatrix}\right)$

$w\begin{bmatrix} 1 \\ 1 \end{bmatrix} + b < w\begin{bmatrix} 0 \\ 1 \end{bmatrix} + b$

Let $w = [w_1 \; w_2]$

$[w_1 \; w_2]\begin{bmatrix} 1 \\ 1 \end{bmatrix} + b < [w_1 \; w_2]\begin{bmatrix} 0 \\ 1 \end{bmatrix} + b$

$w_1 + w_2 + b < w_2 + b$

$w_1 < 0$

Now let's look at for $x_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ $x_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$:

$f(x_1) < f(x_2)$

$w\begin{bmatrix} 0 \\ 0 \end{bmatrix} + b < w\begin{bmatrix} 1 \\ 0 \end{bmatrix} + b$

$b < w_1 + b$

$0 < w_1$

$\Rightarrow$ we have a contradiction, $\therefore$ XOR can not be represented using Linear Model ■

6. $h(x) = w^{(3)} \max\{0, w^{(2)}\{0, w^{(1)}x + b^{(1)}\} + b^{(2)}\} + b^{(3)}$

$w^{(1)} = \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}$ $w^{(2)} = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$ $w^{(3)} = [1 \ 1]$

$b^{(1)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $b^{(2)} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ $b^{(3)} = -1$

$x_0 = 2$

$[1 \ 1] \max\{0, \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \max\{0, \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}[2] + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} - 1$

$= [1 \ 1] \max\{0, \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 3 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} - 1$

$= [1 \ 1] \max\{0, \begin{bmatrix} 7 \\ 8 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} - 1$

$= [1 \ 1]\begin{bmatrix} 7 \\ 9 \end{bmatrix} - 1 = 7 + 9 - 1 = 15$

$\left.\begin{matrix} x = 2 \\ w = 7 \\ b = 1 \end{matrix}\right\} \Rightarrow y = 7x + 1 = 7(2) + 1 = 15$

$\frac{dh}{dx} = w = 7$

$x_0 = -1$

$[1 \ 1] \max\{0, \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \max\{0, \begin{bmatrix} 1.5 \\ 0.5 \end{bmatrix}[-1] + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} - 1$

$= [1 \ 1] \max\{0, \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \max\{0, \begin{bmatrix} -1.5 \\ 0.5 \end{bmatrix}\} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} - 1$

$= [1 \ 1] \max\{0, \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} - 1$

$= [1 \ 1] \max\{0, \begin{bmatrix} 1 \\ 0.5 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}\} - 1$

$= [1 \ 1]\begin{bmatrix} 1 \\ 1.5 \end{bmatrix} - 1 = 1 + 1.5 - 1 = 1.5$

$\left.\begin{matrix} x = -1 \\ w = -1 \\ b = 0.5 \end{matrix}\right\} y = -1x + 0.5 = (-1)(-1) + 0.5 = 1.5$

$\frac{dh}{dx} = w = -1$

$X_0 = 1$

$$[1\ 1]\max\left\{0, \begin{bmatrix}1 & 2\\ 2 & 1\end{bmatrix}\max\left\{0, \begin{bmatrix}1.5\\ 0.5\end{bmatrix}[1] + \begin{bmatrix}0\\ 1\end{bmatrix}\right\} + \begin{bmatrix}0\\ 1\end{bmatrix}\right\} - 1$$

$$= [1\ 1]\max\left\{0, \begin{bmatrix}1 & 2\\ 2 & 1\end{bmatrix}\begin{bmatrix}1.5\\ 1.5\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}\right\} - 1$$

$$= [1\ 1]\max\left\{0, \begin{bmatrix}4.5\\ 4.5\end{bmatrix} + \begin{bmatrix}0\\ 1\end{bmatrix}\right\} - 1$$

$$= [1\ 1]\begin{bmatrix}4.5\\ 5.5\end{bmatrix} - 1 = 4.5 + 5.5 - 1 = 9$$

$x = 1$
$w = 8$
$b = 1$

$y = 8x + 1 = 8(1) + 1 = 9$

$$\frac{dh}{dx} = w = 8$$

7. Given : $f(x) = |x|$ divides input space into 2 regions for value y

For each linear region, the region of the input is a bijection

$$W_{ij}^{(1)} = \begin{cases} 2 & \text{if } i=j \\ 0 & \text{if } i \neq j \end{cases} \leftarrow W \text{ is a diagonal matrix}$$

$$b_i^{(1)} = -1$$

$$f_1(x) = |W^{(1)} x + b|$$

Proof : Take hypercube and partition it into halves with hyperplanes in each dimension then for each region R_i, there is a bijection

$$x_i = \begin{cases} \frac{-y_i+1}{2} & \text{if } y_i = 1, \quad x_i = \{-1, +1\} \\ \frac{+y_i+1}{2} & \end{cases}$$

$$f_i(x_i) = |W^{(1)} x + b^{(1)}| = \sum_j W_{ij} x_i + b_i$$

$$= |2 x_i + b_i|$$

$$= \left|2\left(\frac{\pm y_i + 1}{2}\right) - 1\right|$$

$$= |\pm y_i + 1 - 1| = y_i$$

assume $x_1 \neq x_2$

$\Rightarrow 2x_1 \neq 2x_2$

$\Rightarrow |f(x_1)| \neq |f(x_2)|$

$\Rightarrow$ we have a bijection and $|R| = |(0,1)^d| = 2^d$ possible choices of input regions R_i

Alternatively, we can see that for $d=1$

$0 = (0,1)^1$

input space $R = \{(-1,0), (0,1)\} \Rightarrow$ 2 input regions

then it is intuitive to see that

$0 = (0,1)^d \Rightarrow 2^d$ input regions ∎

8. g has n_g regions of $(0,1)^d$
f has n_f regions of $(0,1)^d$

$f \circ g(\cdot) = f(g(\cdot))$

Let G be input region of $g()$
Let F be input region of $f()$

① Each subregion in G bijects onto f because $g(G)$ bijects onto $(0,1)^d$

② f bijects between F and O

① + ② $\Rightarrow$ $f \circ g$ bijects between G and O

$g(G) \Rightarrow |n_g|$

$f(g(G)) \Rightarrow |n_f n_g|$ regions

9. $h_1 = |W_1 x + b_1|$

$h_2 = |W_2 h_1 + b_2|$

$\vdots$

$h_L = |W_L h_{L-1} + b_L|$

$x \in (0,1)^d \qquad f(x) = h_L$

Each h_i is implicitly a function of x:

$h_1(x) = |W_1 x + b_1|$

$h_2(x) = |W_2(W_1 x + b_1) + b_2|$

$\vdots$

$h_L \circ h_{L-1} \circ \cdots \circ h_1(x) = |W_L h_L \cdots h_1(x) + b_L|$

Proof by induction:

Base case: $L = 1$ we know from q4 that $h_1(x)$ identifies 2^d input region space

$2^{Ld} = 2^{1d} = 2^d$

Inductive Step:

Give true for $L = \ell - 1 \Rightarrow 2^{(\ell-1)d}$ input region space, prove true for $L + 1 = \ell$

Let $g(x) = h_{\ell-1} h_{\ell-2} \cdots h_1(x) + b_{\ell-1}$ identify $2^{(\ell-1)d}$ input region space

Let $f(x) = h_\ell(g(x))$

By q8 we know $f \circ g(\cdot) = |n_f n_g|$ input region space

h_ℓ identifies 2^d input region space

$\Rightarrow f(x) = h_\ell(g(x)) = h_\ell \circ g \Rightarrow |n_{h_\ell} n_g| = |2^d \cdot 2^{(\ell-1)d}| = 2^{\ell d}$ input region space

■

10. $\min_{w \in \mathbb{R}^d} f(w) = \sum_{i=1}^{n} (y_i - w^T x_i)^2$ ①

$$w^{(t+1)} = w^{(t)} - \eta \nabla f(w^{(t)})$$

$$\vdots$$

$$w^{gd} = w^{(gd-1)} - \eta \nabla f(w^{(gd-1)})$$

$$\Downarrow$$

$$w^{gd} = \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \|w\|_2^2 \quad \text{s.t. } Xw = y \quad ②$$

$$\|w\|_2^2 = \sum_i^d w_i^2$$

For ②:

Introduce Lagrange Multipliers

$$L(x, \lambda) = \|w\|_2^2 + x^T (wX - y)$$

s.t. $\nabla_x L = 2X + w^T \lambda = 0$

$\nabla_y L = wX - y = 0$

$$\Rightarrow \lambda = -2(ww^T)^{-1} y$$

$$\Rightarrow X = w^T (ww^T)^{-1} y \quad ③$$

For ①: $w^{gd} = (wX - y)^2 + 2(y - wX)$

$$X = (w^T w + I)^{-1} w^T y$$

$$= w^T (ww^T)^{-1} y \quad ④$$

③ = ④ $\Rightarrow$ solutions found by both are same

11. (25) means soln will have smallest norm of any soln
(24) is the optimization problem
⇒ gradient descent will find the optimal soln for the loss function

12. Least squares regression can always find global min of linear system if it has a soln.

Key contributions
The paper presents findings on double descent, where when model size is increased, performance first decreases then increases. They present empirical evidence that challenges and reconciles some of the conventional wisdoms of statistical theories and machine learning. They show that when model complexity is small compared to the number of samples, test error as a function of model complexity behaves like the classical bias/variance tradeoff. However, when model complexity large enough to interpolate, increasing complexity only decreases test error.

Strengths
The paper thoroughly presents experiments and finds that back their hypothesis and findings such as when the model complexity is at the transition from under to over parameterization, increasing the number of samples shifts the peak to the right and actually increases test error. The paper's findings provide useful ways of thinking and explanation for model complexity and performance.

Weaknesses
Many of the paper's findings cease to exist with optimal early stopping. Thus, models with good early stopping will not find much benefit from the findings of the paper. The paper does not provide a full understanding of optimal early stopping and double descent.

My Takeaways

This fact that most of the paper's findings are negated in the presence of optimal early stopping makes me wonder how to achieve optimal early stopping. With different data and different architectures, optimal early stopping should be related to model complexity in relation to number of samples.

Collaborators:

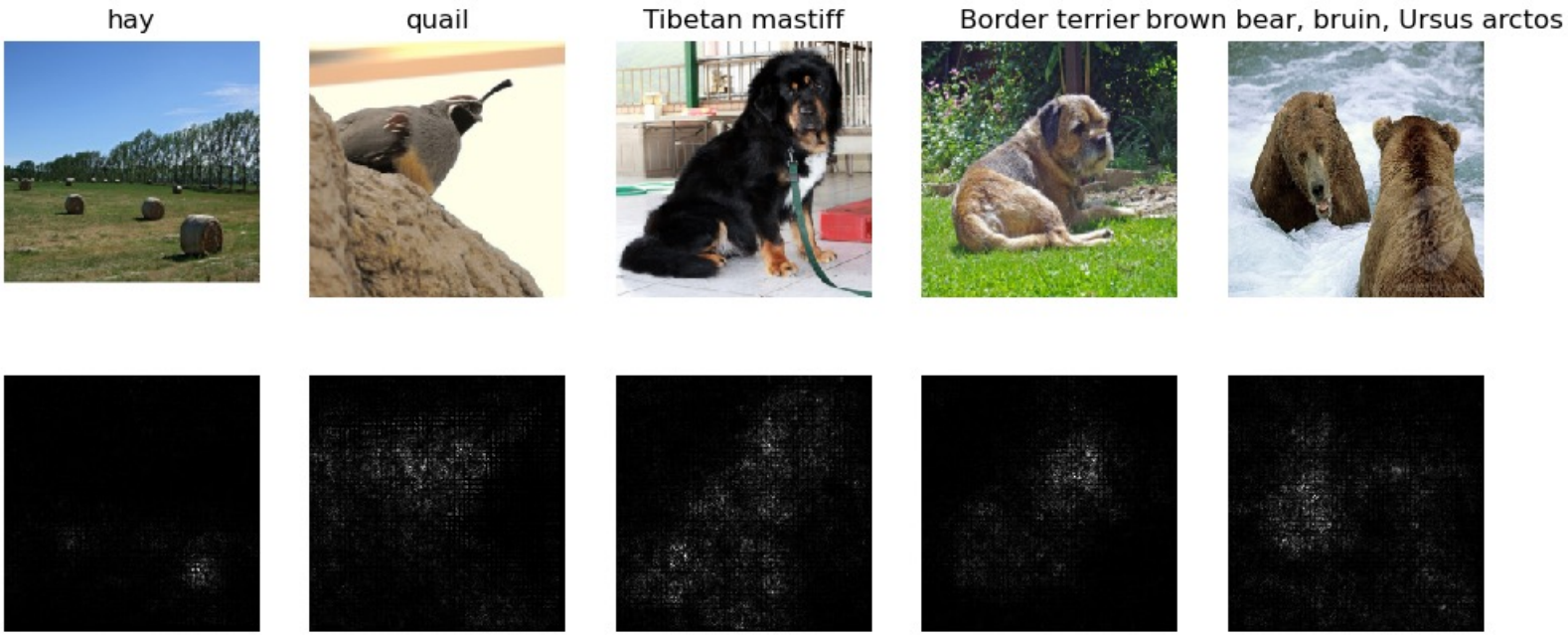
ML Theory textbook

Assignment 3

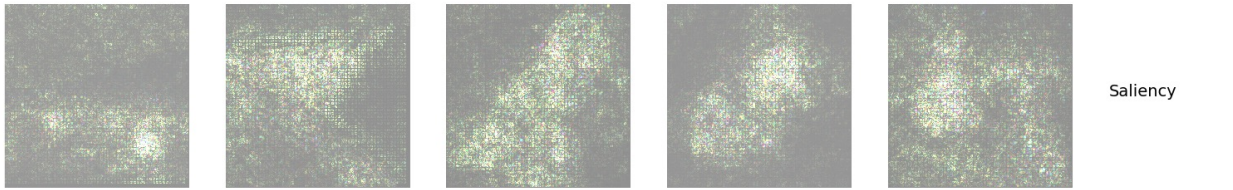
Your name: Bojun Yang
Your GTID: byang301

Visualization

Saliency Map (1 point)



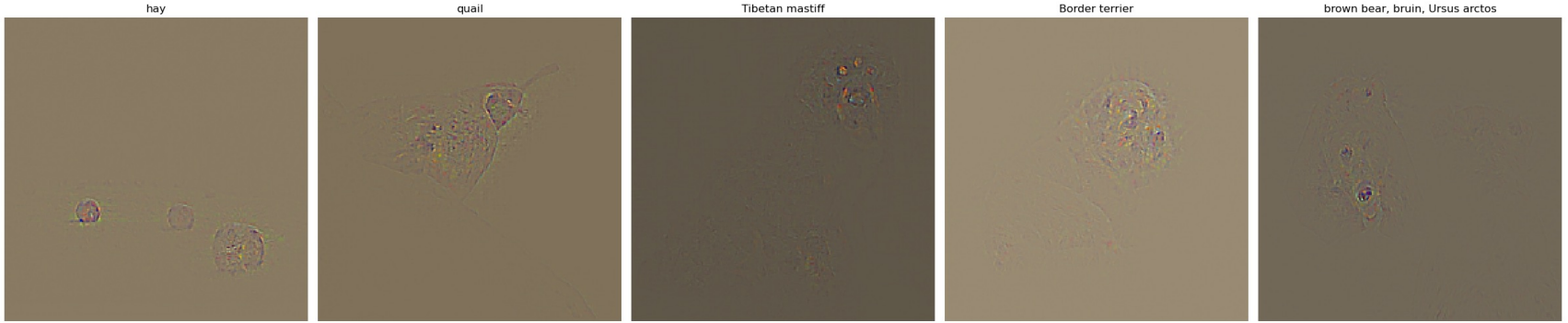
Saliency Map Captum (1 point)



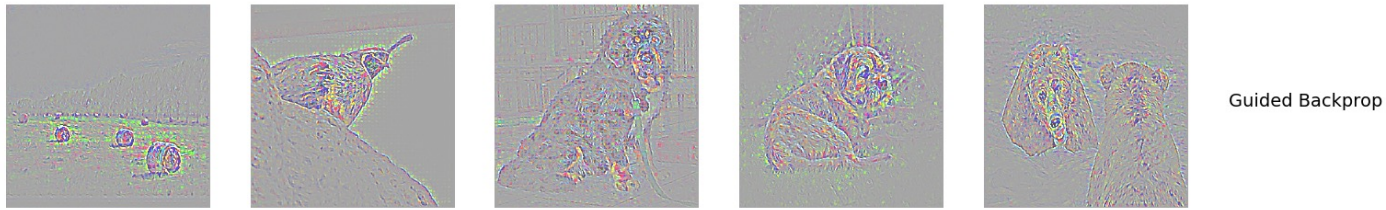
Guided Backprop (1 point)

GradCam (1 point)

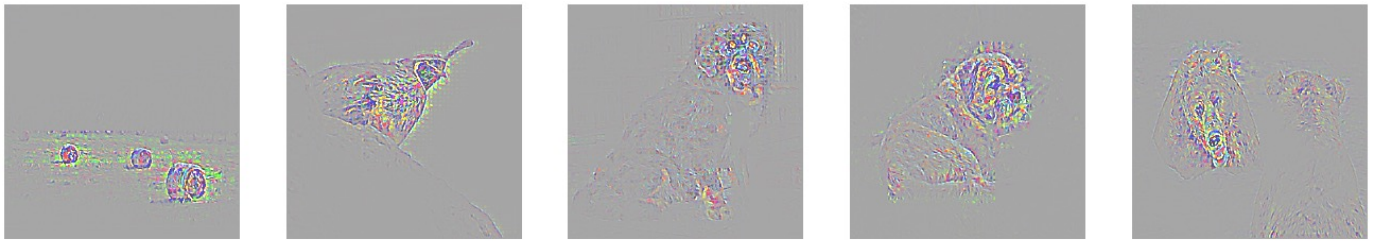
Guided GradCam (1 point)



Guided Backprop + GradCam (Captum) (1 point)

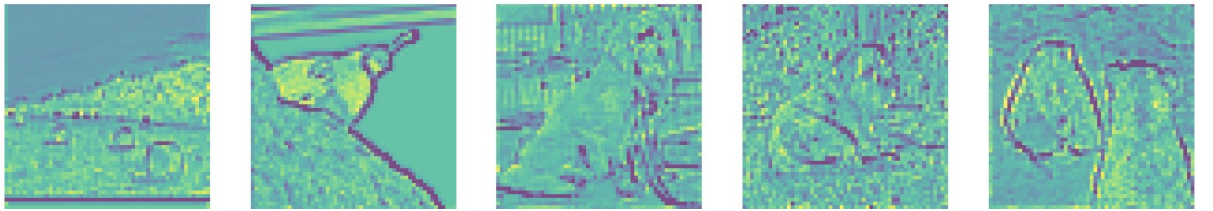



hay
quail
Tibetan mastiff
Border terrier
brown bear, bruin, Ursus arctos
Original Image


Guided GradCam

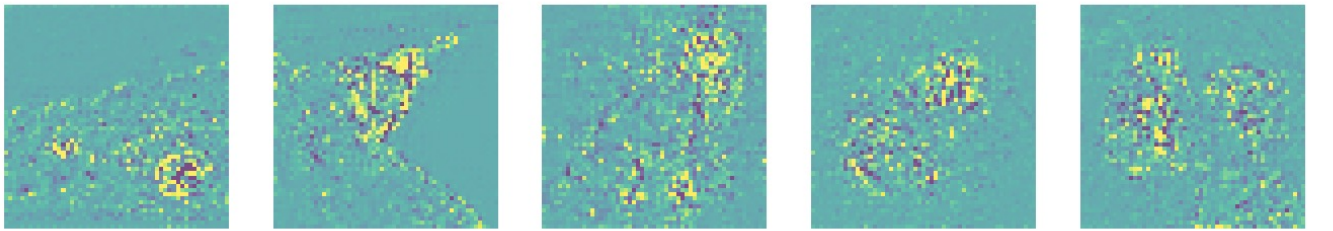
Layers and neurons using Captum (1 point)

Original Image



Layer GradCam

Original Image

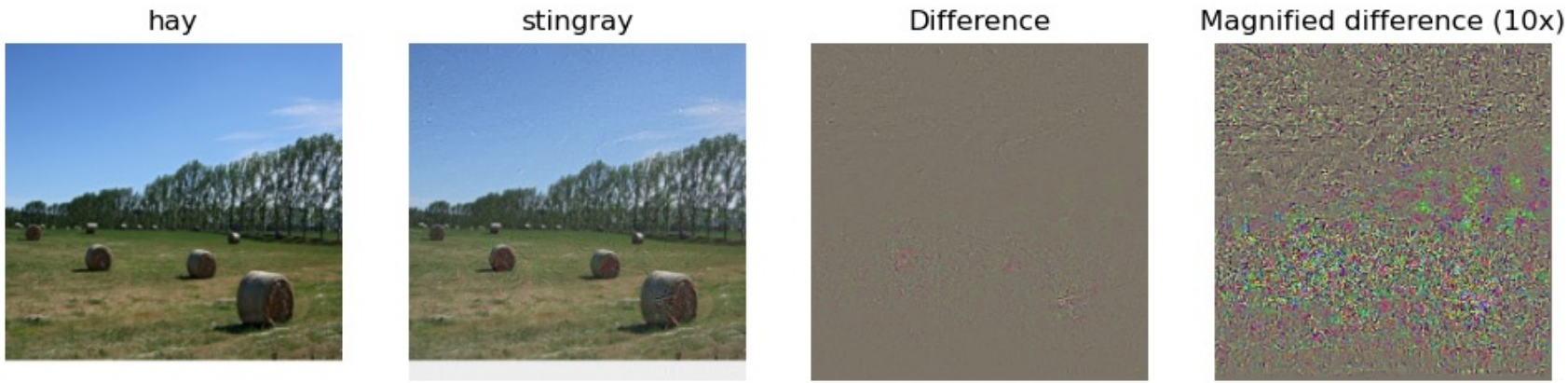


Layer Conductance

What do saliency map and Gradcam tell you? (1 point)

- Saliency map tells use the degree to which each pixel in the image affects the classification score of that image. Brighter pixels correspond to higher effect on the classification.
- Gradcam tells use where the network is looking when it is classifying an input image. It highlights the important regions within the input image for classification.

Fooling Image (1 point)



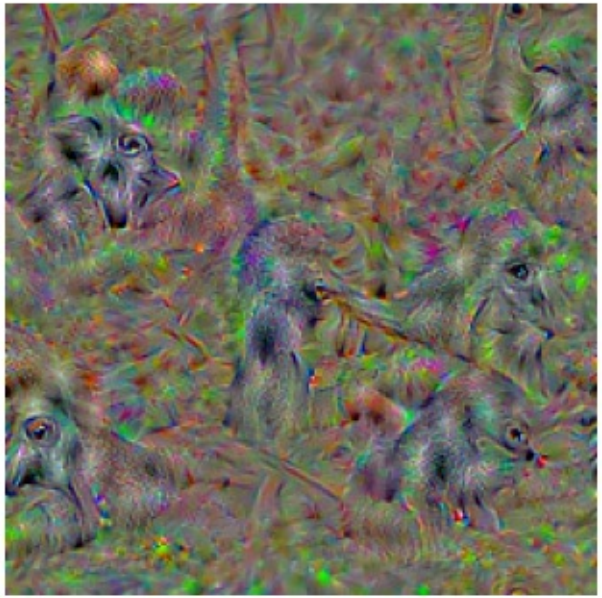
Fooling Image Insights (1 point)

What insights do you get from fooling images:

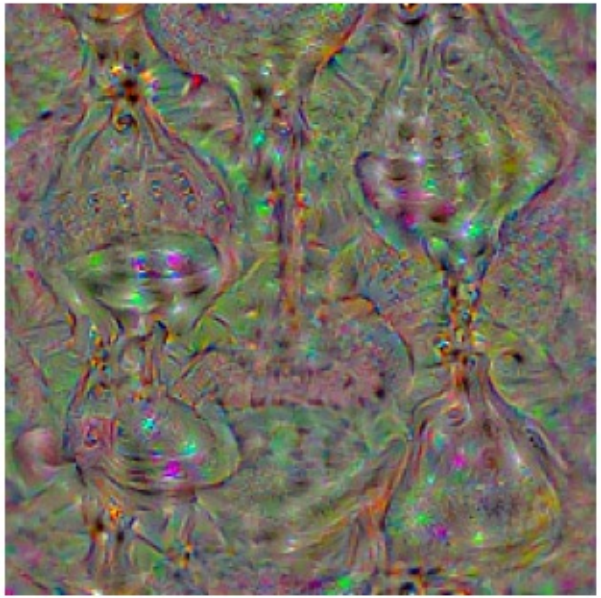
- Even though the fooling image would fool the network into classifying it as a stingray, it still looks the same as hay to the human. This tells us that networks can be easily tricked since it only takes very small differences in the pixels to confuse a nn.

Class Visualization (3 points)

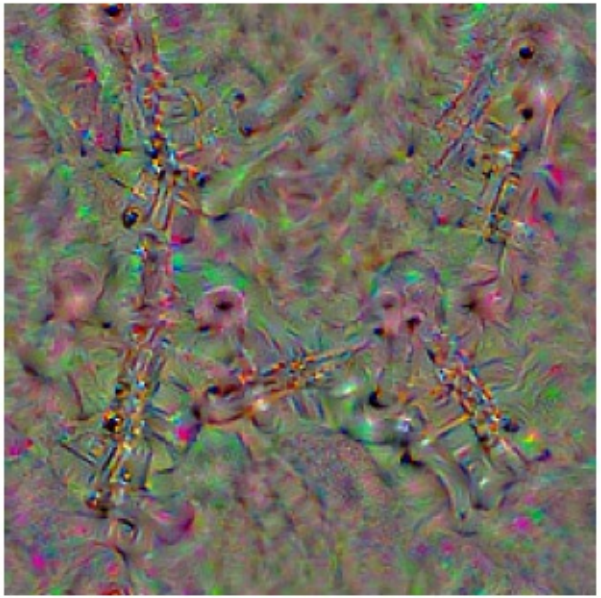
gorilla, Gorilla gorilla
Iteration 100 / 100



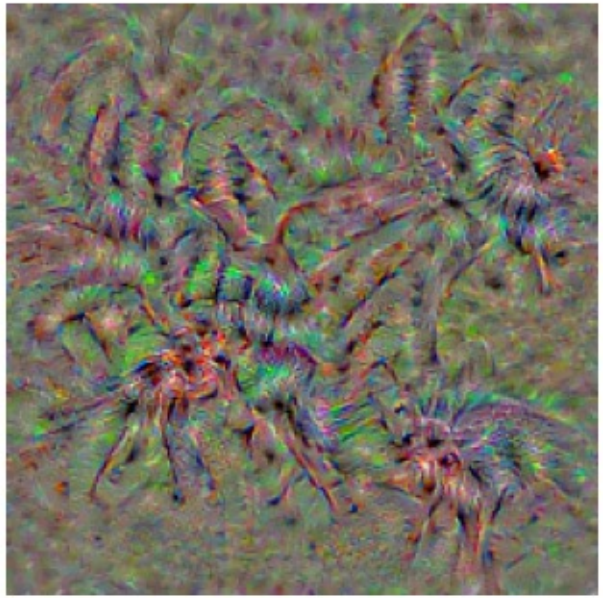
hourglass
Iteration 100 / 100



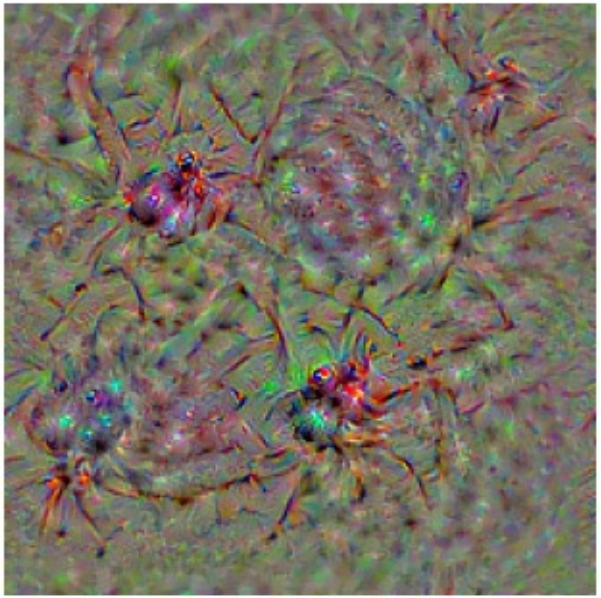
oboe, hautboy, hautbois
Iteration 100 / 100



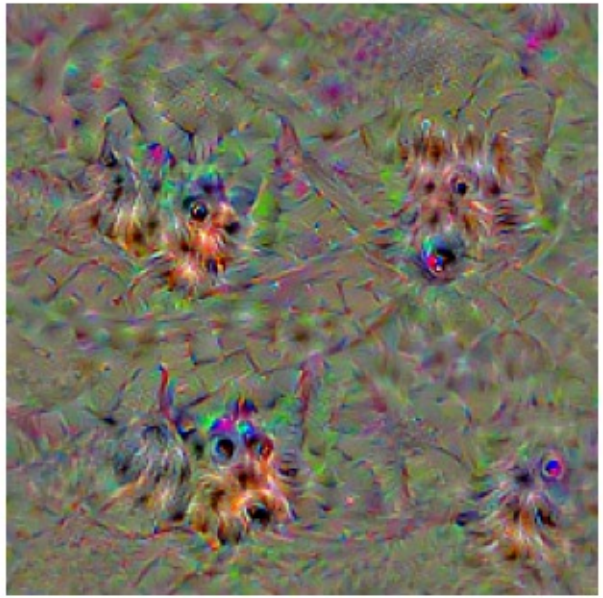
tarantula
Iteration 100 / 100



tick
Iteration 100 / 100



Yorkshire terrier
Iteration 100 / 100



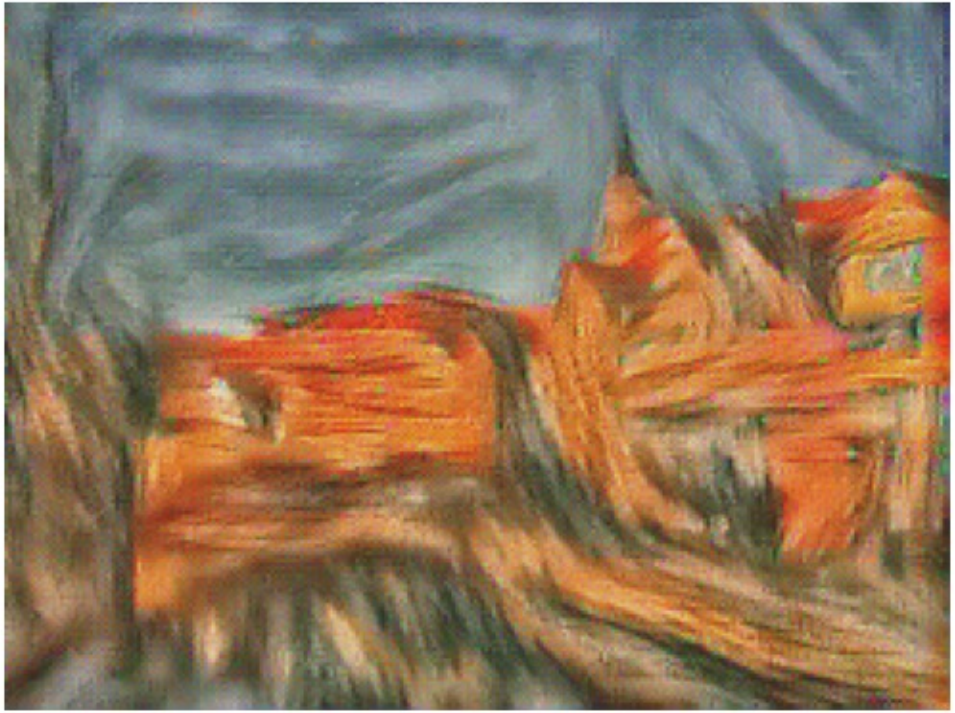
Style Transfer

Composition VII + Tubingen (1 point)

Content Source Img.

Style Source Img.

Scream + Tubingen (1 point)



Content Source Img.

Style Source Img.

Starry Night + Tubingen (1 point)

Content Source Img.

Style Source Img.

- Be sure to append the jupyter notebook i.e. $root/test_style_transfer.ipynb for testing sections of the style transfer implementation to this report.

In [1]:

```
import numpy as np
import torch
from torch.autograd import Variable
import torchvision
import PIL

from style_modules import ContentLoss, StyleLoss, TotalVariationLoss
from style_utils import features_from_img, extract_features, rel_error, style_transfer
from image_utils import preprocess
```

In [2]:

```
def content_loss_test(correct, content_loss):
    content_image = 'styles_images/tubingen.jpg'
    image_size = 192
    content_layer = 3
    content_weight = 6e-2

    c_feats, content_img_var = features_from_img(content_image, image_size, cnn, dtype)

    bad_img = Variable(torch.zeros(*content_img_var.data.size()))
    feats = extract_features(bad_img, cnn)

    student_output = content_loss(content_weight, c_feats[content_layer], feats[content_layer]).data.numpy()
    error = rel_error(correct, student_output)
    print('Content Loss Maximum error is {:.3f}'.format(error))

def gram_matrix_test(correct, style_loss):
    style_image = 'styles_images/starry_night.jpg'
    style_size = 192
    feats, _ = features_from_img(style_image, style_size, cnn, dtype)
    student_output = style_loss.gram_matrix(feats[5].clone()).data.numpy()
    error = rel_error(correct, student_output)
    print('Gram Matrix Maximum error is {:.3f}'.format(error))


def style_loss_test(correct, style_loss):
    content_image = 'styles_images/tubingen.jpg'
    style_image = 'styles_images/starry_night.jpg'
    image_size = 192
    style_size = 192
    style_layers = [1, 4, 6, 7]
    style_weights = [300000, 1000, 15, 3]
```

```
    c_feats, _ = features_from_img(content_image, image_size, cnn, dtype)
    feats, _ = features_from_img(style_image, style_size, cnn, dtype)
    style_targets = []
    for idx in style_layers:
        style_targets.append(style_loss.gram_matrix(feats[idx].clone()))

    student_output = style_loss(c_feats, style_layers, style_targets, style_weights).data.numpy()
    error = rel_error(correct, student_output)
    print('Style Loss Error is {:.3f}'.format(error))


def tv_loss_test(correct, tv_loss):
    content_image = 'styles_images/tubingen.jpg'
    image_size = 192
    tv_weight = 2e-2

    content_img = preprocess(PIL.Image.open(content_image), size=image_size)
    content_img_var = Variable(content_img.type(dtype))

    student_output = tv_loss(content_img_var, tv_weight).data.numpy()
    error = rel_error(correct, student_output)
    print('TV Loss Error is {:.3f}'.format(error))
```

In [3]:

```
dtype = torch.FloatTensor
# Uncomment out the following line if you're on a machine with a GPU set up for PyTorch!
# dtype = torch.cuda.FloatTensor

cnn = torchvision.models.squeezenet1_1(pretrained=True).features
cnn.type(dtype)

# Fix the weights of the pretrained network
for param in cnn.parameters():
    param.requires_grad = False

answers = np.load('data/style-transfer-checks.npz') # for local test
```

Test content loss (1pt)

In [4]:

```
content_loss = ContentLoss()
content_loss_test(answers['cl_out'], content_loss) # should print out 0.0
```

```
Content Loss Maximum error is 0.000
```

Test style loss (2pt)

In [5]:
```
style_loss = StyleLoss()
gram_matrix_test(answers['gm_out'], style_loss) # should print out 0.0
style_loss_test(answers['sl_out'], style_loss) # should print out 0.0
```

```
Gram Matrix Maximum error is 0.000
Style Loss Error is 0.000
```

Test total variation loss (1pt)

In [6]:
```
tv_loss = TotalVariationLoss()
tv_loss_test(answers['tv_out'], tv_loss) # should print out 0.0
```

```
TV Loss Error is 0.000
```