Problem Session 5

Deep Learning and Neural Networks

- Gradient Descent and Backpropagation
- Hyperparameter search
- Wiener Deconvolution + Deep Learning
- See notes for detailed derivations!

Task 1: Backpropagation

- A fancy way of calculating the derivative via the **chain rule**.

$$\mathbf{x} \xrightarrow{\quad f \quad} \mathbf{h} \xrightarrow{\quad g \quad} \mathbf{y} \qquad \frac{\partial \mathbf{y}}{\partial \mathbf{x}}(\mathbf{x}) = \underbrace{\frac{\partial \mathbf{y}}{\partial \mathbf{y}}}_{=1} \cdot \frac{\partial g}{\partial \mathbf{h}}(\mathbf{h}) \cdot \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x})$$

  - h and y are computed in the **forward pass**, derivatives are computed in the **backward pass**

Task 1: Backpropagation

- Forward and backward passes for Linear and ReLU functions
  - Working with row vectors, not column vectors.

- Analytic gradients for gradcheck

- Train a simple network to do image inpainting

Task 1: Gradient of Vector w.r.t. Vector

$$h = gW^T$$

$$\frac{\partial \mathcal{L}}{\partial h_i}$$

$$h_i = \sum_j W_{ij} g_j$$

$$\frac{\partial \mathcal{L}}{\partial g_j} = \sum_i \frac{\partial h_i}{\partial g_j} \frac{\partial \mathcal{L}}{\partial h_i}$$

$$\frac{\partial h_i}{\partial g_j} = W_{ij}$$

$$\frac{\partial \mathcal{L}}{\partial g_j} = \sum_i W_{ij} \frac{\partial \mathcal{L}}{\partial h_i}$$

$$\frac{\partial \mathcal{L}}{\partial g} = \frac{\partial \mathcal{L}}{\partial h} W$$

Takeaway: Gradient is the matrix-vector product of the upstream gradient and the weight matrix

Task 1: Gradient of Vector w.r.t Matrix

$$h = gW^T$$

$$\frac{\partial \mathcal{L}}{\partial h_i}$$

$$h_i = \sum_j W_{ij} g_j$$

$$\frac{\partial \mathcal{L}}{\partial W_{jk}} = \sum_i \frac{\partial h_i}{\partial W_{jk}} \frac{\partial \mathcal{L}}{\partial h_i}$$

$$\frac{\partial h_i}{\partial W_{jk}} = \begin{cases} g_k & i = j \\ 0 & i \neq j \end{cases}$$

$$\frac{\partial \mathcal{L}}{\partial W_{jk}} = \frac{\partial h_j}{\partial W_{jk}} \frac{\partial \mathcal{L}}{\partial h_j}$$

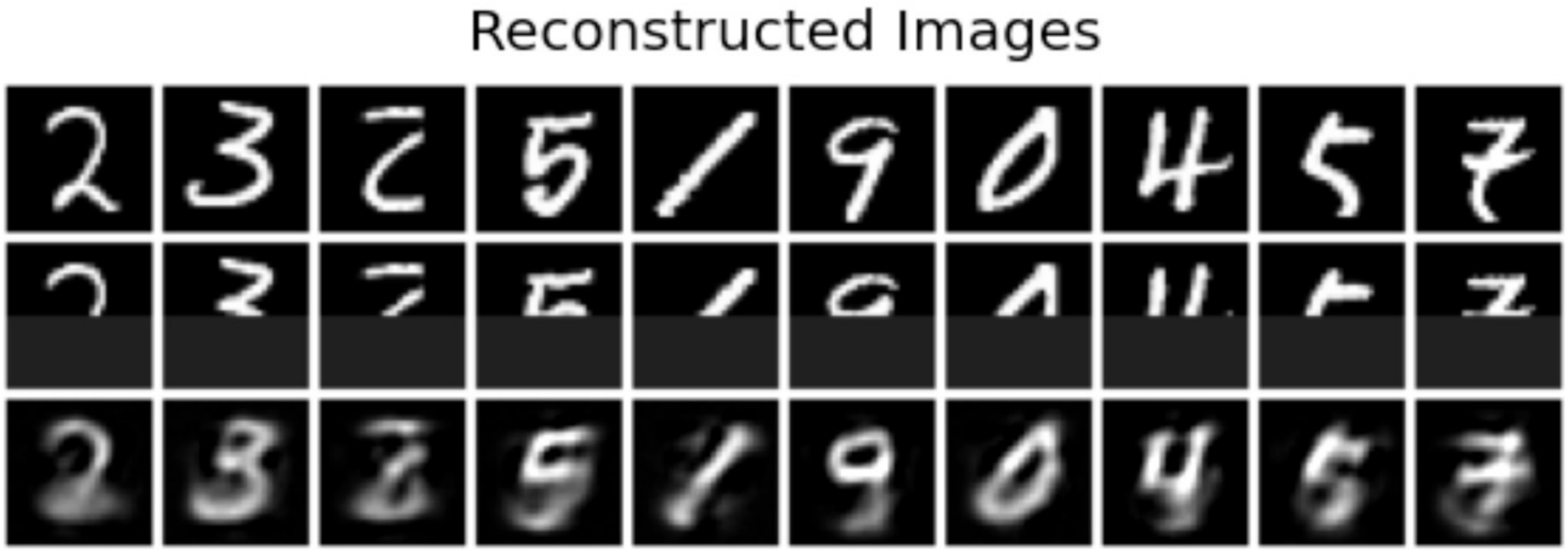
$$\frac{\partial \mathcal{L}}{\partial W} = \left(\frac{\partial \mathcal{L}}{\partial h} \right)^T g$$

Takeaway: Gradient is the outer product of the upstream gradient and the input vector

Task 1: Backpropagation

```
class LinearFunction(Function):

    @staticmethod
    def forward(ctx, input, weight, bias):

        # we will save the input, weight, and bias to help us calculate the
        # gradients in the backward pass
        ctx.save_for_backward(input, weight, bias)

        # return the output of the linear layer
        return input.mm(weight.T) + bias[None, :]

    @staticmethod
    def backward(ctx, grad_output):

        # retrieve the saved variables from the context
        input, weight, bias = ctx.saved_tensors
```

Task 1: Backpropagation



Task 2: Ablation Study

- A tool to evaluate the importance of a component of a neural network.
  - By seeing what happens when we remove it.
- Use a validation dataset to assess if the change works or not
  - Data not seen during training
- Note: Adam Optimizer (usually better than gradient descent)
- Tip: Read the starter code carefully! Most of the stuff is already there.
  - Use train() -> Returns a trained model, given hyperparameter settings
  - Use evaluate_model() -> Evaluates a model on a fixed noise level

Task 2: Ablation Study

		PSNR (dB)		
Uses Bias?	Hidden Channels	$\sigma = 0.01$	$\sigma = 0.1$	$\sigma = 0.2$
✓	32	31.09 	29.47 	22.40 
✓	64	30.97 	**29.73** 	21.41 
✗	32	32.48 	29.20 	24.90 
✗	64	**33.05** 	29.61 	**25.58** 

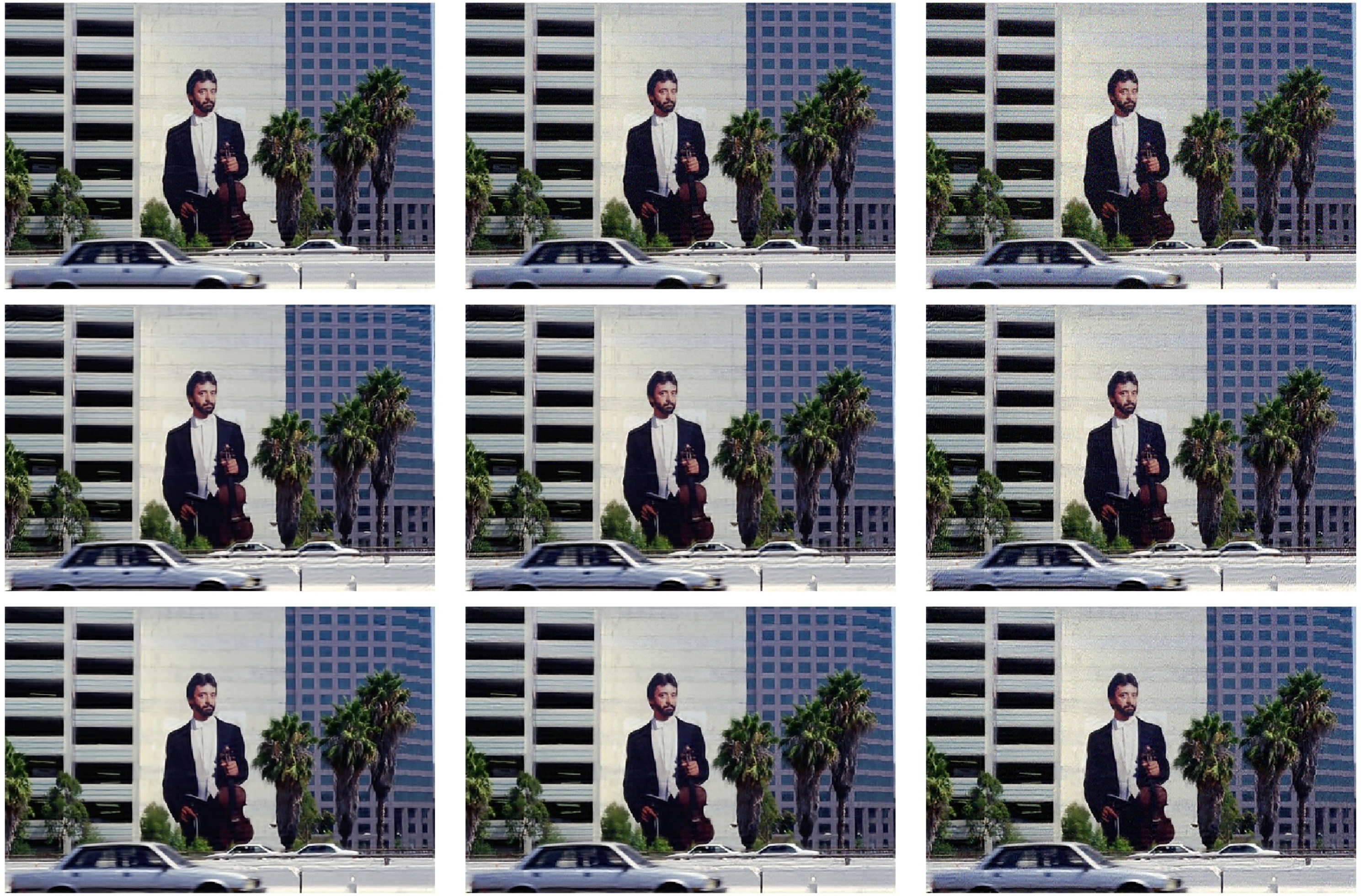
Task 3: Wiener Deconvolution + Deep Learning

- Comparing
  - Wiener Deconvolution only
  - Neural Network only
  - Wiener + Neural Network
- We provide:
  - Wiener deconvolution function (`wiener_deconv()`)
  - Two pretrained neural networks (`load_models()`)
    - One trained to deconvolve and denoise
    - One trained to denoise the output of Wiener deconvolution
- Simulate the noise and apply all three methods to the noisy image!

Task 3: Wiener Deconvolution + Deep Learning

	PSNR (dB)		
	$\sigma = 0.005$	$\sigma = 0.01$	$\sigma = 0.02$
Method 1	**32.16**	28.14	22.86
Method 2	29.17	28.54	25.33
Method 3	30.89	**30.28**	**27.60**

$\sigma = 0.005$ $\sigma = 0.01$ $\sigma = 0.02$

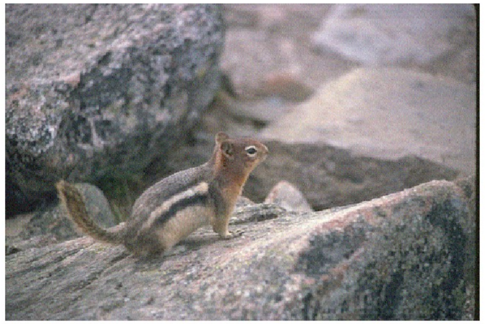


Method 1

Method 2

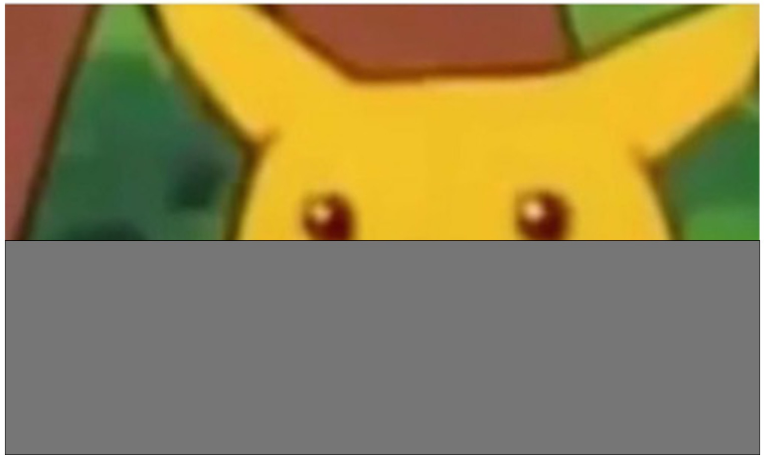
Method 3

Task 3: Wiener Deconvolution + Deep Learning

	σ = 0.005	σ = 0.01	σ = 0.02
Method 1			
Method 2			
Method 3			

Me: *forgets how backpropagation works*

My model:



Good luck with the homework!