PURDUE

BME 64600 – 001 and ECE 60146 – 001

Midterm #2, Spring 2025

NAME _____

PUID _____

Exam instructions:

- You have 75 minutes to work the exam.
- This is a closed-book and closed-note exam. You may not use or have access to your book, notes, any supplementary reference, a calculator, or any communication device including a cell-phone or computer.
- You may not communicate with any person other than the official proctor during the exam.

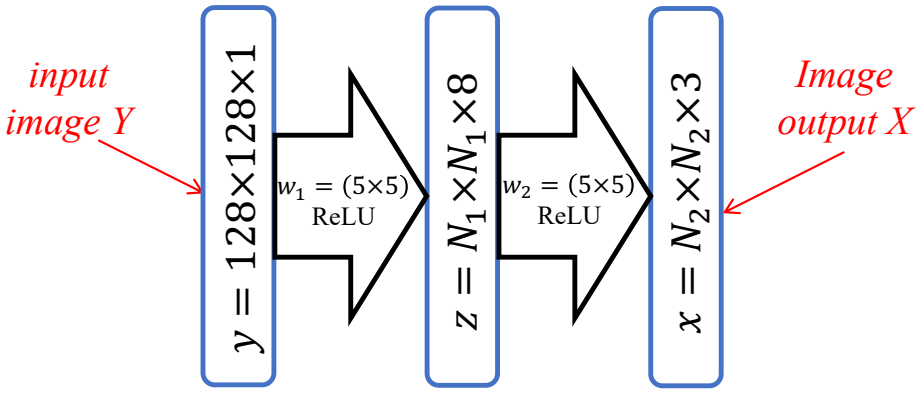
To ensure Gradescope can read your exam:

- Write your full name and PUID above and on the top of every page.
- Answer all questions in the area designated for each problem.
- Write only on the front of the exam pages.
- DO NOT run over to the next question.

Name/PUID: _____

Problem 1. (30pt) Parameters of Convolutional Neural Networks

Consider the following convolutional neural network pictured below with a gray-scale image input and a color image output. Each layer uses a ReLu activation function, a **"valid"** boundary condition, and denote the convolution kernels by w_1 and w_2 and the associated offsets by b_1 and b_2.



1a) Calculate the value of N_1.

Solution: $N_1 = 124$

1b) Calculate the value of N_2.

Solution: $N_2 = 120$

1c) Calculate the shape of w_1.

Solution: $5 \times 5 \times 1 \times 8$

1d) Calculate the shape of w_2.

Solution: $5 \times 5 \times 8 \times 3$

1e) Calculate the shape of b_1.

Solution: 8

1f) Calculate the total number of parameters in the model.

Solution: Number of parameters in each layer:

- layer1:
  - filter: $5 \times 5 \times 1 \times 8 = 200$
  - offset: 8
- layer2:
  - filter: $5 \times 5 \times 8 \times 3 = 600$
  - offset: 3

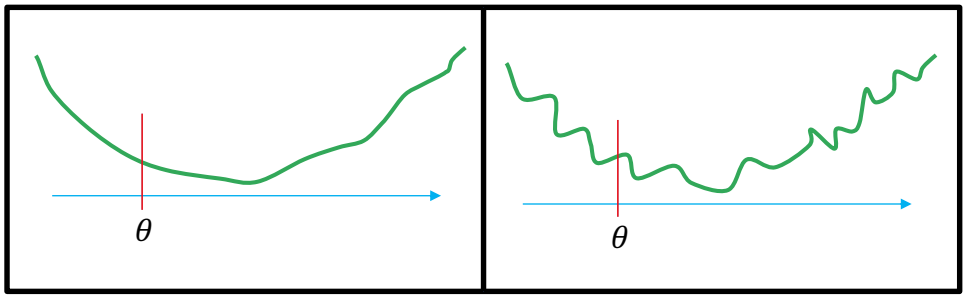
Total number of parameters:

$$200 + 8 + 600 + 3 = 811$$

Name/PUID: _____

Problem 2. (15pt) Batch Size

Consider the following "smooth" function and "bumpy" function illustrated below.



Smooth Function Bumpy Function

2a) Which function has more local minima?

Solution: The bumpy function has more local minima

2b) For which function is it better to use a **large** batch size? Why?

Solution: The large batch size is better for the smooth function because it reduces that random variation in the gradient. This in turn allow the global minimum to be more precisely computed in the final "exploitation" stage of optimization.

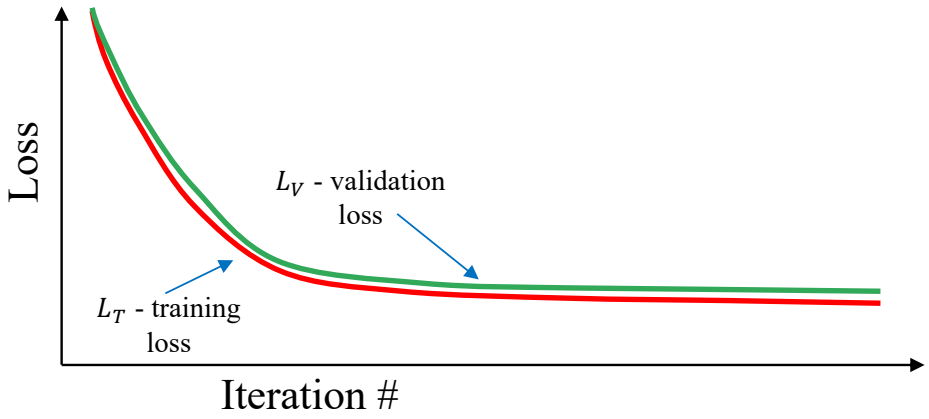
2c) For which function is it better to use a **small** batch size? Why?

Solution: The small batch size is better for the bumpy function because for two reasons: a) It generates more noise in the gradient that helps jump out of local minima. This improves the "exploration" portion of the optimization in the early stages of optimization; b) It speeds evaluation of the gradient, so the algorithm can perform more gradient steps per unit time.

Name/PUID: _____

Problem 3. (10pt) Training Convergence

Consider the following convergence functions.



3a) What is the data telling you?

Solution: This is telling you that even after many iterations, the training loss is not significantly better than the validation loss. This means two things: a) There is sufficient training data for the capacity of the model; b) There model may not have sufficient capacity to best solve the ML problem.

3b) What actions should you take to improve your results? Why?

Solution: Based on this it would be good to increase the capacity of the model by, for example, increasing the number of layers, increasing the size of the convolution kernel, adding blocks to the networks that make it more expressive.

Name/PUID: _____

Problem 4. (35pt) Back Propagation

A neural network can be represented by

$$x = f(y) = Ay + b \ ,$$

where $y \in \mathbb{R}^N$ is the input, $x \in \mathbb{R}^N$, $A \in \mathbb{R}^{N \times N}$, $b \in \mathbb{R}^N$, and $N = 4$. Furthermore, assume that

$$A_{i,j} = w_{i-j} \ ,$$

for some function w_i. Also define the loss function

$$L(y) = \frac{1}{2K} \sum_{k=0}^{K-1} \|x_k - f(y)\|^2 \ ,$$

where $\{x_k\}_{k=0}^{K-1}$ are K training samples.

4a) Write out the matrix A in terms of w_i.

Solution:

$$A = \begin{bmatrix} w_0 & w_{-1} & w_{-2} & w_{-3} \\ w_1 & w_0 & w_{-1} & w_{-2} \\ w_2 & w_1 & w_0 & w_{-1} \\ w_3 & w_2 & w_1 & w_0 \end{bmatrix}$$

4b) Describe in words the meaning of the operation Ay.

Solution: The operation Ay corresponds to convolution of

$$y = [y_0, y_1, y_2, y_3] \ ,$$

with the kernel

$$w = [w_{-3}, w_{-2}, w_{-1}, w_0, w_1, w_2, w_3] \ ,$$

using circular convolution with the "same" boundary condition.

4c) Write out an explicit expression for the adjoint gradient of the function $[\nabla_y f(y)]^t$ in terms of A.

Solution:

$$[\nabla_y f(y)]^t = [A]^t = A^t$$

4d) Write out the matrix A^t in terms of w_i.

Solution:

$$A = \begin{bmatrix} w_0 & w_1 & w_2 & w_3 \\ w_{-1} & w_0 & w_1 & w_2 \\ w_{-2} & w_{-1} & w_0 & w_1 \\ w_{-3} & w_{-2} & w_{-1} & w_0 \end{bmatrix}$$

4e) Describe in words the meaning of the operation $[\nabla_y f(y)]^t \epsilon$.

Solution: Since

$$[\nabla_y f(y)]^t \epsilon = A^t \epsilon \ ,$$

we see that this represents convolution of

$$\epsilon = [\epsilon_0, \epsilon_1, \epsilon_2, \epsilon_3] \ ,$$

with the kernel

$$w = [w_3, w_2, w_1, w_0, w_{-1}, w_{-2}, w_{-3}] \ ,$$

using circular convolution with the "same" boundary condition.

4f) Write out an explicit expression for the adjoint gradient of the loss function $[\nabla_y L(y)]^t$ in terms of A, x_k, and $f(y)$.
(Hint, this should result in a column vector.)

Solution:

$$\begin{aligned}
[\nabla_y L(y)]^t &= \left[\nabla_y \frac{1}{2K} \sum_{k=0}^{K-1} \|x_k - f(y)\|^2 \right]^t \\
&= \left[\frac{1}{2K} \sum_{k=0}^{K-1} \nabla_y \|x_k - f(y)\|^2 \right]^t \\
&= \left[\frac{1}{K} \sum_{k=0}^{K-1} [x_k - f(y)]^t A \right]^t \\
&= \frac{1}{K} \sum_{k=0}^{K-1} A^t [x_k - f(y)] \\
&= A^t \frac{1}{K} \sum_{k=0}^{K-1} [x_k - f(y)]
\end{aligned}$$

4g) Describe in words how you evaluate the function, $[\nabla_y L(y)]^t$.

Solution: First you average all the prediction errors $[x_k - f(y)]$ for each training sample. Then you perform a convolution with the time reverse kernel w_{-i} using "same" boundary conditions.

Name/PUID: _____

Problem 5. (20pt) Bayes Classifier

Let $Y \in \mathbb{R}^N$ be a random vector whose distribution depends on the class, C. Moreover, C is a binary random variable such that $P\{C = 0\} = \pi_0$ and $P\{C = 1\} = 1 - \pi_0$.
Then when $C = 0$, the distribution is given by $Y \sim p_0(y)$, and when $C = 1$, the distribution is given by $Y \sim p_1(y)$.
Also define the likelihood ratio

$$R(y) = \frac{p_1(y)}{p_0(y)} \ .$$

5a) Calculate the conditional distribution of Y given C denoted by $p(y|c)$.

Solution:

$$p(y|c) = \begin{cases} p_o(y) & \text{if } c = 0 \\ p_1(y) & \text{if } c = 1 \end{cases}$$

5b) Calculate $p(y)$, the marginal distribution of Y.

Solution:

$$p(y) = p_0(y)\pi_0 + p_1(y)(1 - \pi_0)$$

5c) Calculate the conditional probability of $P\{C = 0|Y = y\} = f(y)$ in terms of the likelihood ratio $R(y)$.

Solution:

$$\begin{aligned} f(y) &= P\{C = 0|Y = y\} \\ &= \frac{p_0(y)\pi_0}{p_0(y)\pi_0 + p_1(y)\pi_1} \\ &= \frac{\pi_0}{\pi_0 + R(y)\pi_1} \end{aligned}$$

5d) Specify an expression for the Bayes classifier, $B(y)$, that classifies Y as either $C = 0$ or $C = 1$ with the minimum probability of classification error in terms of $f(y)$.

Solution: The Bayes classifier is given by

$$B(y) = \begin{cases} 0 & \text{if } f(y) \geq 1/2 \\ 1 & \text{if } f(y) < 1/2 \end{cases}$$

Name/PUID: _____

Problem 6. (25pt) Nash Equilibrium

Consider the two loss functions for the generator and discriminator in a GAN defined below

$$g(\theta_g, \theta_d)$$
$$d(\theta_g, \theta_d) \ ,$$

where g and d are loss functions, and θ_g and θ_d are the parameters of the generator and discriminator, respectively. Next define the following function

$$D(\theta_g) = \arg\min_{\theta_d} d(\theta_g, \theta_d) \ .$$

6a) Write the equations for the Nash equilibrium using the two loss functions g and d.

Solution:

$$\theta_g^* = \arg\min_{\theta_g} g(\theta_g, \theta_d^*)$$
$$\theta_d^* = \arg\min_{\theta_d} d(\theta_g^*, \theta_d)$$

6b) Give an interpretation for the function $D(\theta_g)$.

Solution: The function $D(\theta_g)$ determines the best discriminator parameter, θ_d^*, for each choice of generator parameter, θ_g.

6c) Substitute the function $D(\theta_g)$ into the generator loss function and write a single equation for the Nash equilibrium solution for the generator parameter.

Solution:

$$\theta_g^* = \arg\min_{\theta_g} g(\theta_g, D(\theta_g))$$

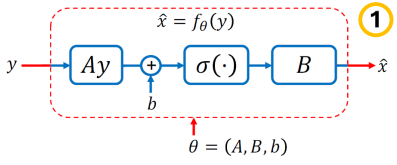
6d) What do we call the special case when $g(\theta_g, \theta_d) = -d(\theta_g, \theta_d)$?

Solution: Zero-sum game

6e) Give an example of an activity people engage in that can be modeled as a Nash equilibrium with $g(\theta_g, \theta_d) = -d(\theta_g, \theta_d)$.

Solution: Any game, such as chess or checkers or go, in which only one play can win is a zero-sum game.

Single Layer NN Graphically: 1



Where: $f_\theta(y) = B\sigma(Ay + b)$

M-dimensional simplex:

$$S^M = \left\{ x \in \Re^M : \forall i, x_i \geq 0 \text{ and } 1 = \sum_{i=0}^{M-1} x_i \right\}$$

Softmax: $\sigma_i(z) = \dfrac{e^{z_i}}{\sum_j e^{z_j}}$

Gradient of the Softmax:

$$[\nabla\sigma(z)]_{i,j} = \frac{1}{\sum_k e^{z_k}} \left(e^{z_i}\delta_{i-j} - \frac{e^{z_i} e^{z_j}}{\sum_k e^{z_k}} \right)$$

MSE Loss:

$$L_{MSE}(\theta) = \frac{1}{K} \sum_{k=0}^{K-1} \|x_k - f_\theta(y_k)\|^2$$

Gradient Descent and line search: 2

Repeat until converged {
$d \leftarrow -\nabla L(\theta)$
$\alpha^* \leftarrow \arg\min_\alpha \{L(\theta + \alpha d^t)\}$
$\theta \leftarrow \theta + \alpha^* d^t$
}

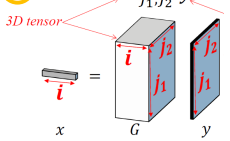
Steepest Descent Algorithm

Loss gradient (MSE):

$$-\nabla_\theta L_{MSE}(\theta) = \frac{2}{K} \sum_{k=0}^{K-1} \left(x_k - f_\theta(y_k)\right)^t \nabla_\theta f_\theta(y_k)$$

Einstein Notation: 3

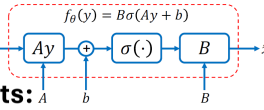
$$x^i = G^i{}_{j_1,j_2}\, y^{j_1,j_2}$$



$$\delta^i{}_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$$

$$G^k{}_j = G^k{}_i\, \delta^i{}_j$$

Delta Functions:



Gradients:

w.r.t. y $\left[\nabla_y f\right]^i{}_j = B^i{}_{i_1} [\nabla\sigma]^{i_1}{}_{i_2} A^{i_2}{}_j$

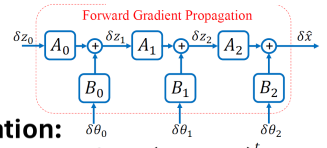
w.r.t. A $[\nabla_A f]^{i_1}{}_{j_1,j_2} = B^{i_1}{}_{i_2} [\nabla\sigma]^{i_2}{}_{j_1}\, y^{j_2}$

w.r.t. b $[\nabla_b f]^{i_1}{}_{j_1} = B^{i_1}{}_{i_2} [\nabla\sigma]^{i_2}{}_{j_1}$

w.r.t. B $[\nabla_B f]^{i_1}{}_{j_1,j_2} = \delta^{i_1}{}_{j_1} \sigma^{j_2}$

Adjoint of the Loss Gradient: 4

$$[\nabla L_{MSE}(\theta)]^t = \frac{-2}{K} \sum_{k=0}^{K-1} [\nabla f_\theta(y_k)]^t \left(x_k - f_\theta(y_k)\right)$$



Back Propagation:

$$g_{m,k} = \nabla_{\theta_m} \|x_k - f_\theta(y_k)\|^2 = -2\left(x_k - f_\theta(y_k)\right)^t \nabla f_\theta(y_k)$$

$$\nabla_{\theta_m} L(\theta) = g_m = \frac{1}{K} \sum_{k=0}^{K-1} g_{m,k}$$ → *Gradient of Loss Function*

MAE: $\|\mathrm{a} - \mathrm{b}\|_1^1 = \sum_i |a_i - b_i|$

MC Cross Entropy: $-\sum_i \mathrm{a}_i \log b_i$

Back Prop for Loss Functions:

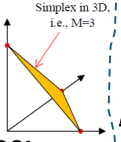
MSE: $\epsilon_k^t = -2\left(x_k - f_\theta(y_k)\right)$

MAE: $\epsilon_k^t = -sign\left(x_k - f_\theta(y_k)\right)$

CE: $\epsilon_k^t = -\dfrac{x_k}{f_\theta(y_k)}$

Discrete Probability Density:

$$P\{X \in A\} = \sum_{i \in A} p_i$$



Expected, Mean, Variance:

$$\mu = E[X] = \sum_{n=0}^{M-1} n\, p_n$$ *Expected Value of X*

$$E[f(X)] = \sum_{n=0}^{M-1} f(n)\, p_n$$ *Expected Value of f(X)*

$$\sigma^2 = E[(X-\mu)^2] = \sum_{n=0}^{M-1} (n-\mu)^2 p_n$$ *Variance X*

Conditional Density:

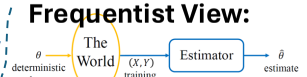
$$p_{x|y}(i|j) = P\{X = i | Y = j\} = \frac{P\{X = i, Y = j\}}{P\{Y = j\}} = \frac{p(i,j)}{p_y(j)} = \frac{p(i,j)}{\sum_{i=0}^{M-1} p(i,j)}$$

$$p_{y|x}(j|i) = P\{Y = j | X = i\} = \frac{P\{X = i, Y = j\}}{P\{X = i\}} = \frac{p(i,j)}{p_x(i)} = \frac{p(i,j)}{\sum_{j=0}^{M-1} p(i,j)}$$

Conditional Expectation:

$$E[f(X)|Y] = \sum_{n=0}^{M-1} f(n)\, p_{x|y}(n|Y) \quad E[f(Y)|X] = \sum_{n=0}^{M-1} f(n)\, p_{y|x}(n|X)$$

Frequentist View: 6



MLE:

$$\hat{\theta} = \arg\max_\theta \{p_\theta(X,Y)\}$$

Regression

$$= \arg\min_\theta \{-\log p_\theta(X,Y)\}$$

$$\hat{\theta} = \arg\min_\theta \{-\log p_\theta(x,y)\}$$

$$= \arg\min_\theta \{-\log p_\theta(x|y) - \log p(y)\}$$

$$= \arg\min_\theta L_{MSE}(\theta; x, y)$$

Classification (w/ 1-hot encoding)

$$\hat{\theta} = \arg\min_\theta \{-\log p_\theta(x,y)\}$$

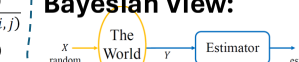
$$= \arg\min_\theta \{-\log p_\theta(x|y) - \log p(y)\}$$

$$= \arg\min_\theta \{L_{CE}(\theta; x, y)\}$$

Multinomial Distro

$$\hat{\theta} = \arg\min_\theta \left\{ -\sum_{i=0}^{M-1} \frac{N_i}{N} \log \theta_i \right\} = \frac{N_i}{N}$$

Bayesian View:



Bayes Law:

$$p_{x|y}(x|y) = \frac{p_{y|x}(y|x)\, p_x(x)}{p_y(y)} = \frac{p_{y|x}(y|x)\, p_x(x)}{\sum_{k=0}^{M-1} p_{y|x}(y|k)\, p_x(k)}$$

Min Mean Square Error

$$\hat{X} = E[X|Y] = \sum_{x=0}^{M-1} x\, p_{x|y}(x|Y)$$

MAP Estimate

$$= \arg\min_x \{-\log p_{y|x}(y|x) - \log p_x(x)\}$$

Bias and Variance

Frequentist

$\text{bias}_\theta = E[\hat{\theta}|\theta] - \theta$

$\text{Var}_\theta = E\left[\left\|\hat{\theta} - E[\hat{\theta}|\theta]\right\|^2 |\theta\right]$

$\text{MSE}_\theta = \text{Var}_\theta + (\text{bias}_\theta)^2$

Bayesian

$\text{bias}_x = E[\hat{X}|X = x] - x$

$\overline{\text{bias}^2} = \text{E}\left[\left\|E[\hat{X}|X] - X\right\|^2\right]$

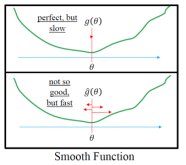
$\text{Var} = E\left[\left\|\hat{X} - E[\hat{X}|X]\right\|^2\right]$

$\text{MSE} = \text{Var} + \overline{\text{bias}^2}$

Stochastic Gradient Descent: 8

$$\hat{g} = g + \frac{w}{\sqrt{K_b}}$$

Batch Gradient, True Gradient, Noise



SGD w/ Momentum:

init $v \leftarrow 0$

Repeat until converged {
Repeat $b = 1$ to B {
$d \leftarrow -\nabla L(\theta; S_b)$
$v \leftarrow \gamma v + \alpha(1-\gamma)d$
$\theta \leftarrow \theta + v^t$

ADAM Optimizer:

Momentum+Preconditioning

init $v \leftarrow 0$; $r \leftarrow 0$;
init $t \leftarrow 0$
Repeat until converged {
Repeat $b = 1$ to B {
$t \leftarrow t + 1$
$d \leftarrow -\nabla L(\theta; S_b)$
$v \leftarrow \beta_1 v + (1-\beta_1)d$
$r \leftarrow \beta_2 r + (1-\beta_2)d^2$
$\hat{v} \leftarrow v/(1-\beta_1^t)$
$\hat{r} \leftarrow r/(1-\beta_2^t)$
$\theta \leftarrow \theta + \alpha\left(\sqrt{\hat{r}} + \epsilon\right)^{-1} \hat{v}$

Batch 9

Input parameters:
ϵ – small number to regularize division
α – momentum parameter
Learned parameters:
γ, β – scaling and offset parameters
Estimated parameters:
μ_l, σ_l^2 – learned mean and variance parameters
Parameters saved for inference:
μ_l, σ_l^2 – learned mean and variance parameters
γ, β, ϵ – same as above

Training

Initialize $\mu_l \leftarrow 0$; $\sigma_l^2 \leftarrow v$
For each b (batch) {
$\mu_B \leftarrow \frac{1}{K_b} \sum_{k \in S_b} z_k$
$\sigma_B^2 \leftarrow \frac{1}{K_b - 1} \sum_{k \in S_b} (z_k - \mu_B)^2$
For each $k \in S_b$ {
$\hat{z}_k \leftarrow \frac{z_k - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$
$y_k \leftarrow \gamma \hat{z}_k + \beta$
}
$\mu_l \leftarrow \alpha\, \mu_l + (1-\alpha)\mu_B$
$\sigma_l^2 \leftarrow \alpha\, \sigma_l^2 + (1-\alpha)\sigma_B^2$

Inference

puts:
ϵ – same constant as used in training
γ, β – learned from training
μ_l, σ_l^2 – estimated during training
For each input {
$\hat{z} \leftarrow \frac{z - \mu_l}{\sqrt{\sigma_l^2 + \epsilon}}$
$y \leftarrow \gamma \hat{z} + \beta$

Denoising:

$$\hat{x} = E[x|y] = E[y - w|y] = y - E[w|y] = y - f_{\hat{\theta}}(y)$$

where $\hat{w} = f_{\hat{\theta}}(y)$

Positional Encoding:

$$P(t, 2i) = \sin\left(\frac{2\pi}{\lambda^i} t\right)$$

$$P(t, 2i+1) = \cos\left(\frac{2\pi}{\lambda^i} t\right)$$

Convolution "Same" Boundary: 5

$$x(j_1, j_2) = \sum_{k_1=-1}^{1} \sum_{k_2=-1}^{1} z(j_1 - k_1, j_2 - k_2)\, w(k_1, k_2)$$

or $x^{j_1,j_2} = w_{(j_1,j_2)} * z^{(j_1,j_2)}$

Convolution "Valid" Boundary:

$$x(j_1, j_2) = \sum_{k_1=-1}^{1} \sum_{k_2=-1}^{1} z(j_1 + 1 - k_1, j_2 + 1 - k_2)\, w(k_1, k_2)$$

or $x^{j_1,j_2} = w_{(j_1,j_2)} * z^{(j_1,j_2)}$

2D Convo w/ vector input: D_i=2, D_o=1

$$x(j_1, j_2) = \sum_{i=0}^{d-1} \sum_{k_1=-p}^{p} \sum_{k_2=-p}^{p} z(j_1 - k_1, j_2 - k_2, i)\, w(k_1, k_2, i)$$

or $x^{j_1,j_2} = w_{(j_1,j_2),i} * z^{(j_1,j_2),i}$

2D Convo w/ vector input/output:

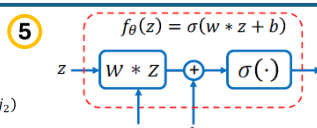
D_i=2, D_o=2 $x^{j_1,j_2,j_3} = w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i}$

CNN Average Pooling: *d=stride*

$$x(j_1, j_2) = \sum_{k_1=0}^{p-1} \sum_{k_2=0}^{p-1} \frac{1}{p^2} z(d_1 j_1 + k_1, d_2 j_2 + k_2)$$

CNN Max Pooling:

$$x(j_1, j_2) = \max_{0 \leq k_1 < p} \max_{0 \leq k_2 < p} z(d_1 j_1 + k_1, d_2 j_2 + k_2)$$



w.r.t. input:

Forward Function

$f(y) = \sigma\left(w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i} + b^{j_3}\right)$

Adjoint Gradient

$\delta_{j_1,j_2,i} = w_{(-j_1,-j_2),i}{}^{j_3} [\nabla\sigma]^{k_1,k_2,k_3}{}_{(j_1,j_2),j_3} \epsilon_{k_1,k_2,k_3}$

w.r.t. b:

Forward Function

$f(y) = \sigma\left(w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i} + b^{j_3}\right)$

Adjoint Gradient

$[g_b]_{j_3} = \mathbf{1}^{j_1,j_2} [\nabla\sigma]^{k_1,k_2,k_3}{}_{j_1,j_2,j_3} \epsilon_{k_1,k_2,k_3}$

w.r.t. w:

Forward Function

$f(y) = \sigma\left(w_{(j_1,j_2),i}{}^{j_3} * z^{(j_1,j_2),i} + b^{j_3}\right)$

Adjoint Gradient

$[g_w]_{j_1,j_2,i,j_3} = z^{(-j_1,-j_2),i} [\nabla\sigma]^{k_1,k_2,k_3}{}_{(j_1,j_2),j_3} \epsilon_{k_1,k_2,k_3}$

Update Directions:

Gradient step: $A \leftarrow A + \alpha d^t$ → *Replace A with b or B*

For A

$$d_{j_1,j_2} = -\nabla_A L(\theta) = \frac{2}{K} \sum_{k=0}^{K-1} \epsilon_{k,i_1} B^{i_1}{}_{i_2} [\nabla\sigma_k]^{i_2}{}_{j_1} y_k{}^{j_2}$$

For b

$$d_{j_1} = -\nabla_b L(\theta) = \frac{2}{K} \sum_{k=0}^{K-1} \epsilon_{k,i_1} B^{i_1}{}_{i_2} [\nabla\sigma_k]^{i_2}{}_{j_1}$$

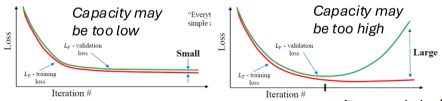
For B

$$d_{j_1,j_2} = -\nabla_B L(\theta) = \frac{2}{K} \sum_{k=0}^{K-1} \epsilon_{k,j_1} \sigma_k{}^{j_2}$$

Data partitions: 7

S_T is the training data
S_V is the validation data
S_E is the testing (evaluation) data

Each partition has the same loss function but different "ground truth" data



Overfitting Fixes:

Regularization – adds penalties to weights, encouraging them to be small or collapse to 0

$S(\theta) = -\log p(\theta)$ ⟸ *MAP estimate*

$S(\theta) = \|\theta\|^2$ ⟸ *Like a Gaussian Prior Reduces amplitude of weights*

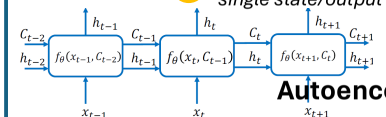
$S(\theta) = \|\theta\|_1$ ⟸ *Like a Laplacian Prior Encourages weights to go to zero*

Dropout – randomly drop nodes in network

For each batch{
For each layer l {
$r^{(l)} \leftarrow Bernoulli(p, shape(y^{(l)}))$
For $n = 0$ to $K_b - 1$ {
For each layer l {
$\tilde{y}^{(l)} \leftarrow r^{(l)} * y^{(l)}$ *Dropout*
$z^{(l+1)} \leftarrow w^{(l+1)} * \tilde{y}^{(l)}$
$y^{(l+1)} \leftarrow f(z^{(l+1)})$

RNN - LSTM: 10

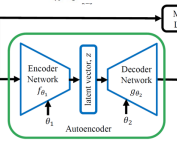
Combines C and h into single state/output



GRU:

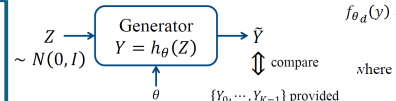
Autoencoder:

$$Loss(\theta) = \sum_{k=0}^{K-1} \left\| y_k - g_{\theta_2}\left(f_{\theta_1}(y_k)\right) \right\|^2$$



Generative Models:

Goal: Generate random vectors that "look" like ground truth



and ideas:

Discriminator Loss Function:

$$\hat{d}(\theta_g, \theta_d) = \frac{1}{K} \sum_{k=0}^{K-1} \left\{ -\log f_{\theta_d}(y_k) - \log\left(1 - f_{\theta_d}(\tilde{y}_k)\right) \right\}$$

G/D Loss Function

$$g(\theta_g, \theta_d) = E\left[-\log f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right]$$

$$d(\theta_g, \theta_d) = E\left[-\log f_{\theta_d}(Y)\right] + E\left[-\log\left(1 - f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right]$$

$$\theta_g^* = \arg\min_{\theta_g} g(\theta_g, \theta_d^*)$$

$$= \arg\min_{\theta_g} E\left[-\log\left(f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right]$$

$$\theta_d^* = \arg\min_{\theta_d} d(\theta_g^*, \theta_d)$$

$$= \arg\min_{\theta_d} \left\{ E\left[-\log f_{\theta_d}(Y)\right] + E\left[-\log\left(1 - f_{\theta_d}\left(\tilde{Y}_{\theta_g}\right)\right)\right] \right\}$$

Log Likelihood Ratio:

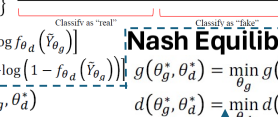
$$f_{\theta_d}(y) \approx P\{Class = R|y\}$$

$$= \frac{p_r(y)}{p_r(y) + p_{\theta_g}(y)} = \frac{1}{1 + R_{\theta_g}(y)}$$

where

$$R_{\theta_g}(y) = \frac{p_{\theta_g}(y)}{p_r(y)}$$

11



Nash Equilibrium:

$$g(\theta_g^*, \theta_d^*) = \min_{\theta_g} g(\theta_g, \theta_d^*)$$

$$d(\theta_g^*, \theta_d^*) = \min_{\theta_d} d(\theta_g^*, \theta_d)$$

2 Agents: Zero Sum:

$$d(\theta_g^*, \theta_d^*) = \max_{\theta_g} d(\theta_g, \theta_d^*)$$

$$d(\theta_g^*, \theta_d^*) = \min_{\theta_d} d(\theta_g, \theta_d^*)$$