VIR 2022 **Name:** _____

Training questions for midterm test

Variant: A Points _____

1. **MLE loss**: You are given probability distribution model $p(y|x, w) = xw \exp(-xwy)$, which models probability of variable $y \in \mathbb{R}^+$, given measurement $x \in \mathbb{R}$ and unknown model parameters $w \in \mathbb{R}$. You are given a training set $\mathcal{D} = \{(x_1, y_1) \dots (x_N, y_N)\}$. Write down the optimization problem, which corresponds to the maximum likelihood estimate of the model parameters w? Simplify resulting optimization problem if possible.

2. **MLE loss**: You are given Laplace probability distribution model

$$p(y|x,w) = \frac{1}{2}\exp(-|wx + b - y|),$$

which models probability of variable $y \in \mathbb{R}^+$, given measurement $x \in \mathbb{R}$ and unknown model parameters $w, b \in \mathbb{R}$. You are given a training set $\mathcal{D} = \{(x_1, y_1) \dots (x_N, y_N)\}$. Write down the optimization problem, which corresponds to the maximum likelihood estimate of the model parameters w, b? Simplify resulting optimization problem if possible.

3. **Simple backpropagation:** Consider the following network

$$y = \Big(\sin(\mathbf{w})\Big)^\top \mathbf{x},$$

where $\sin(\mathbf{w})$ denotes element-wise function $[\sin(\mathbf{w}_1), \sin(\mathbf{w}_2)]^\top$. Consider an input $\mathbf{x} = [2, 1]^\top, \mathbf{w} = [\pi/2, \pi]^\top$ and label $l = 1$.

- Draw the computational graph of the forward pass of this network. Preserve vectorized form, so that the vector $\mathbf{w}$ correspond to a single edge in the graph.

- Compute the forward pass of the network in the computational graph.

- Use loss $\mathcal{L}(y, l) = -\log(y \cdot l)$ to compute the loss value between the forward prediction y and label l. Add this loss to the computational graph.

- Populate the computation graph by local gradients and use the chain rule to compute the gradient $\frac{\partial \mathcal{L}(y,l)}{\partial \mathbf{w}}$ and estimate an update of parameters $\mathbf{w}$ with learning rate $\alpha = 0.5$.

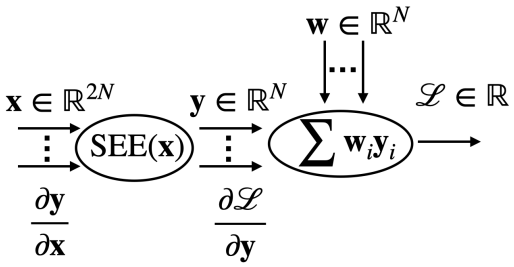
4. **Vector-jacobian-product:** Consider computational graph below, which consists of

   1. function that Select Even Elements (SEE) from input vector $\mathbf{x} \in \mathbb{R}^{2N}$ and returns them as $\mathbf{y} \in \mathbb{R}^{N}$:

   $$\mathbf{y} = \text{SEE}(\mathbf{x})$$

   2. loss function defined as weighted sum

   $$\mathcal{L} = \sum_i \mathbf{w}_i \mathbf{x}_i$$



- What is SEE Jacobian $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$? What is its dimensionality?

- Derive partial derivative of the loss $\mathcal{L}$ wrt $\mathbf{x}$

- Define vector-jacobian-product function $\text{vjp}_{\text{SEE}}(\mathbf{p})$, where $\mathbf{p} \in \mathbb{R}^{\mathbf{N}}$ is assumed to be the upstream gradient vector. Discuss computational and memory requirements on the backward pass using Jacobian multiplications and vjp_{SEE} function.

5. **Conv2D feedforward pass:** In all questions, assume stride denotes length of convolutional stride, pad denotes symmetric zero-padding, rate is dilatation rate of atrous convolution. conv stands for convolution or atrous convolution layer and max is max-pooling layer.

   You are given input feature map (image) $\mathbf{x}$ and kernel $\mathbf{w}$:

   $$\mathbf{x} = \begin{array}{|c|c|c|}\hline 1 & 0 & 2 \\ \hline 2 & 1 & -1 \\ \hline 0 & 0 & 2 \\ \hline \end{array} \qquad \mathbf{w} = \begin{array}{|c|c|}\hline 1 & -1 \\ \hline 0 & 2 \\ \hline \end{array}$$

   Compute outputs of the following layers:

   - $\text{conv}(\mathbf{x},\ \mathbf{w},\ \text{stride} = 1,\ \text{pad} = 0,\ \text{rate} = 2) =$

   - $\text{conv}(\mathbf{x},\ \mathbf{w},\ \text{stride} = 3,\ \text{pad} = 1,\ \text{rate} = 1) =$

   - $\max(\mathbf{x},\ 2 \times 2) =$

6. **Conv1D backward pass:** You are given network (without loss layer) which consists of the convolutional layer and the max-pooling layer. The structure is defined as follows:

$$f(\mathbf{x}, \mathbf{w}) = \max\Big(\text{conv}(\mathbf{x},\ \mathbf{w}),\ 1 \times 2\Big)$$

- Draw computational graph and compute the feed-forward pass for input feature map (image) $\mathbf{x}$ and convolutional kernel $\mathbf{w}$:

$$\mathbf{x} = \boxed{2}\boxed{1}\boxed{2} \quad \mathbf{w} = \boxed{1}\boxed{0}$$

- Estimate gradient wrt kernel $\mathbf{w}$ (i.e. compute local gradients for the max-pooling layer and the convolutional layer and substitute edge-values computed in the feed-forward pass).

$$\frac{\partial f(\mathbf{x}, \mathbf{w})}{\partial \mathbf{w}} =$$

- Update weights using pure SGD ($\alpha = 0.5$).

- Update weights using SGD ($\alpha = 0.5$).

7. **Conv2D dimensionality:** You want to design a convolutional layer which maps $100 \times 100 \times 3$ input (height$\times$width$\times$depth) on $40 \times 40 \times 6$ output feature map. Propose suitable number of kernels, kernel size, stride and padding.

   number_of_kernels =

   kernel_size_h =

   kernel_size_w =

   kernel_size_d =

   stride =

   pad =

8. **Conv2D dimensionality:** You are given an input volume X of dimension $[batch \times channel \times width \times height] = [4 \times 2 \times 13 \times 13]$
   Consider a one 2D convolutional filter F of size $[width \times height] = [5 \times 5]$

   - Assuming a stride of 2, What is the size of padding, which ensures that the feature map is size 7x7 of the input map? Note: A padding size of 1 for a $[30 \times 30]$ image gives it a resulting size of $[32 \times 32]$, in other words, zeros are added on both sides.

   - Calculate the total memory in bytes of the learnable parameters of the filter, assuming that each weight is a dual-precision float (FP64).

   - Calculate the amount of operations performed by a single application of the filter (just one "stamp"). Each addition or multiplication counts as a single operation.
     For example: $\alpha x + \beta y + c$ amounts to 2 multiplication and 2 addition operations, totaling 4 operations.

   - Considering the entire input dimensions of X, given a stride of 2, no padding, calculate the amount of filter applications ("stamps") that you have to perform to process the entire input.

9. **Activation functions:** Discuss and possibly derive advantages and drawbacks of activation functions (sigmoid, relu, lrelu, elu).

10. **Theoretical properties of Conv2D:** You are given convolutional network $y = f(\mathbf{x}, \mathbf{w}, \mathbf{v})$ consisting of two layers

    1. convolutional layer with one 3×3 kernel (stride 1, padding=0), with weights denoted $\mathbf{w}$ (bias is completely ignored for simplicity)
    2. convolutional layer with one 3×3 kernel (stride 1, padding=0), with weights denoted $\mathbf{v}$ (bias is completely ignored for simplicity)

    - What is the dimensionality of input $\mathbf{x}$ if output y is a scalar value?

    - **Hard question (should be considered only for students targeting on A-grade)** Let us assume that you initialized values of all kernels by *zeros*, such that $\mathbf{w} = \mathbf{0}$ and $\mathbf{v} = \mathbf{0}$. You are given a training set consisting of pairs of real-valued, finite inputs $\mathbf{x}_i$ and corresponding real-valued scalar outputs y_i. You trained the network on the training set to minimize L_2-loss using Stochastic Gradient Descent (SGD). What relations (if any) will hold among trained weights after the training (assuming that the SGD converged to a finite values)? Proove your claims if possible.
      Hint: Recall that, if you have convolutional layer $z = \text{conv2d}(\mathbf{x}, \mathbf{w})$ followed by another layer $u = p(z)$, then gradient $\frac{\partial p(\text{conv2d}(\mathbf{x},\mathbf{w}))}{\partial \mathbf{w}} = \text{conv2d}(\mathbf{x}, \frac{\partial p(\mathbf{z})}{\partial \mathbf{z}})$, where $\frac{\partial p(\mathbf{z})}{\partial \mathbf{z}}$ denotes the upstream gradient. Similarly $\frac{\partial p(\text{conv2d}(\mathbf{x},\mathbf{w}))}{\partial \mathbf{x}} = \text{conv2d}(\frac{\partial p(\mathbf{z})}{\partial \mathbf{z}}, \mathbf{w})$ with padding corresponding to a desired gradient size. Look at backprop in computational graph.

- Let us assume, that we create new network $g(\mathbf{x}, \mathbf{w}_1, \dots \mathbf{w}_4, \mathbf{v}_1, \dots \mathbf{v}_4)$ by introducing additional kernels into convolutional layers of $f(\mathbf{x}, \mathbf{w}, \mathbf{v})$. What kind of function is g? Can you replace g by a simplier function $\hat{g}(\mathbf{x}, \theta)$, that preserves *expressing power* of g: i.e. given any parameters $\mathbf{w}_1, \dots \mathbf{w}_4, \mathbf{v}_1, \dots \mathbf{v}_4$, there exists lower dimensional parameter θ such that

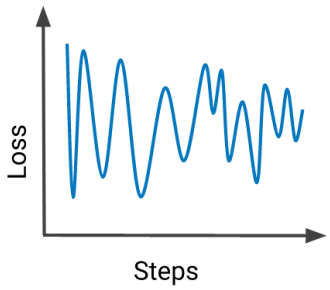
  $$g(\mathbf{x}, \mathbf{w}_i, \mathbf{v}_j) = \hat{g}(\mathbf{x}, \theta)$$

  for any possible $\mathbf{x}$. What lowest possible dimensionality of θ?

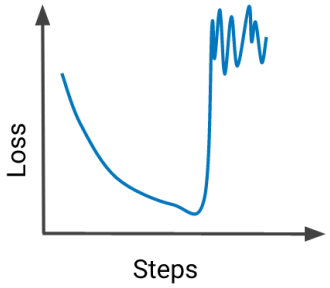
11. You are given a following figures of loss function over the training iterations.

    - Explain for each of the figures, what might be happening to the model during the training based on these curves
    - Propose at least a one way how to solve each of these issues.

1)



2)



3)