Laissez des traces de votre démarche pour toutes les questions !
For all questions, show your work ! Be as brief and specific as possible !

1. **(10 point) Convolutional Neural Nets.**
Consider a CNN with 6 layers applied to an input image of size 256×256. For each layer we perform convolutions with kernels of size 5×5, a 2×2 zero-padding and a stride of 1.
   (a) What are the dimensions of the feature maps for each layer ?

   (b) What is the effective receptive field for the neurons in the 6th layer (i.e. what dimensions of the input does a middle neuron in the feature map "see") ?

   (c) How much zero-padding would you require to make this a "full" convolution ?

   (d) How much zero-padding would you require to make this a "valid" convolution ?

   (e) How much zero-padding would you require to make this a "same" convolution ?

   [FRANÇAIS] Considérez un CNN à six couches appliqué à une image de taille 256×256. Pour chacune des couches, nous performons des convolutions avec des noyaux de taille 5×5, un zéro padding de 2×2 et un 'stride' de 1.
   (a) Quelles sont les dimensions des *features maps* pour chacune des couches ?

   (b) Quel est le *effective receptive field* pour les neurones de la sixième couche (i.e. quelles sont les dimensions de l'image d'entré qu'un neurone peut "voir") ?
   (c) Combien de zéro padding auriez-vous besoin pour effectuer une *full convolution* ?

   (d) De combien de zéro padding auriez-vous besoin pour effectuer une *valid convolution* ?

   (e) Combien de zéro padding auriez-vouss besoin pour effectuer une *same convolution* ?

2. **(15 point) Recurrent Neural Networks.**
Consider the behavior of a linear RNN :

$$\boldsymbol{h}_t = W\boldsymbol{h}_{t-1} + U\boldsymbol{x}_t + \boldsymbol{b},$$

where $\boldsymbol{h}_t \in \mathbb{R}^N$ and $\boldsymbol{x}_t \in \mathbb{R}^M$.
   (a) Write $\boldsymbol{h}_t$ as a function of $\boldsymbol{h}_0$.

   (b) Derive an expression for the Jacobian J :

$$\begin{bmatrix} \frac{\partial \boldsymbol{h}_{t,1}}{\partial \boldsymbol{h}_{0,1}} & \cdots & \frac{\partial \boldsymbol{h}_{t,1}}{\partial \boldsymbol{h}_{0,N}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{h}_{t,N}}{\partial \boldsymbol{h}_{0,1}} & \cdots & \frac{\partial \boldsymbol{h}_{t,N}}{\partial \boldsymbol{h}_{0,N}} \end{bmatrix}$$

(c) What happens when $t \to \infty$? Under what conditions ?

[FRANÇAIS] Considérez le RNN linéaire suivant :

$$\boldsymbol{h}_t = W\boldsymbol{h}_{t-1} + U\boldsymbol{x}_t + \boldsymbol{b},$$

où $\boldsymbol{h}_t \in \mathbb{R}^N$ et $\boldsymbol{x}_t \in \mathbb{R}^M$.

(a) Exprimez $\boldsymbol{h}_t$ en fonction de $\boldsymbol{h}_0$.

(b) Trouvez une expression pour la Jacobienne J :

$$\begin{bmatrix} \frac{\partial \boldsymbol{h}_{t,1}}{\partial \boldsymbol{h}_{0,1}} & \cdots & \frac{\partial \boldsymbol{h}_{t,1}}{\partial \boldsymbol{h}_{0,N}} \\ \vdots & \ddots & \vdots \\ \frac{\partial \boldsymbol{h}_{t,N}}{\partial \boldsymbol{h}_{0,1}} & \cdots & \frac{\partial \boldsymbol{h}_{t,N}}{\partial \boldsymbol{h}_{0,N}} \end{bmatrix}$$

(c) Que ce passe-t-il lorsque $t \to \infty$? Sous quelles conditions ?

3. **(10 point) Optimization.**
Briefly compare and contrast the following pairs of optimization algorithms (be specific) :

(a) Adagrad and RMSprop,

(b) RMSprop and Adam.

[FRANÇAIS] Comparez et contrastez brièvement les paires d'algorithmes d'optimisation suivantes (soyez spécifique !) :

(a) Adagrad et RMSprop,

(b) RMSprop et Adam.

4. **(10 point) Regularization.**

(a) Consider training a neural network with stochastic gradient descent. Explain why the capacity of a neural network grow as the number of training iterations increases ?

[FRANÇAIS] Considirez un réseau de neurones entraîné avec la descente de gradient stochastique. Expliquez pourquoi la capacité du modèle augmente avec chaque itération d'entraînement.

(b) Consider an alternative regularization method to dropout where, for a given example, instead of using a fixed probability of dropping out a neuron from the network, we *learn* the dropout probability. More specifically, we imagine neuron i as having its own dropout probability parameter p_i and we updating these dropout parameters via gradient descent (assuming we have solved the problem of estimating gradients for this parameter) in order to minimize the training loss.

(i) Is this likely that this method would be an effective regularizer ? What pitfalls do you foresee, if any ?

(ii) Above we mentioned that we have solved the problem of estimating gradients for the dropout parameters. What is the problem ?

[FRANÇAIS] Considérez une méthode de régularisation similaire à *dropout*, où, pour un exemple donné, nous *apprenons* la probabilité de *dropout* au lieu d'utiliser une valeur fixe. Plus précisément, un neurone i aurait sa propre probabilité de *dropout* p_i. Ces paramètres seraient mis-à-jour par descente de gradient (en assument que nous savons comment calculer le gradient de ces paramètres) afin de minimiser le coût d'entrainement.

(i) Est-il probable que cette méthode soit un régularisateur efficace ? Le cas échéant, qu'est-ce qui pourrait mal se passer ?

(ii) Plus haut, nous avons assumé que nous savions comment calculer le gradient des paramètres *dropout* p_i. Quel problème devions-nous surmonter afin d'y arriver ?

5. **(10 point) CAE and Weight Decay**

(a) For the case of a linear autoencoder with a squared error loss function, prove that the contractive autoencoder (CAE) penalty is equivalent to weight decay. Specifically, for the linear auto-encoder of data $\boldsymbol{x}$, assume the reconstruction is $\tilde{\boldsymbol{x}} = W^\top \boldsymbol{h}(\boldsymbol{x})$, where $\boldsymbol{h}(\boldsymbol{x}) = W\boldsymbol{x}$ and the CAE loss function is : $L = (\boldsymbol{x} - \tilde{\boldsymbol{x}})^\top(\boldsymbol{x} - \tilde{\boldsymbol{x}}) + \lambda\|\nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})\|_F^2$.

[FRANÇAIS] Prouvez que dans le cas d'un auto-encodeur linéaire accompagné d'une fonction de coût quadratique, la pénalité contractante est équivalente au *weight decay*. Dans votre réponse, présumez que la reconstruction de $\boldsymbol{x}$ est donnée par $\tilde{\boldsymbol{x}} = W^\top \boldsymbol{h}(\boldsymbol{x})$, où $\boldsymbol{h}(\boldsymbol{x}) = W\boldsymbol{x}$, et que la fonction de coût du CAE est $L = (\boldsymbol{x}-\tilde{\boldsymbol{x}})^\top(\boldsymbol{x}-\tilde{\boldsymbol{x}})+\lambda\|\nabla_{\boldsymbol{x}}\boldsymbol{h}(\boldsymbol{x})\|_F^2$.

(b) With the equivalence above thus established, describe if or how the CAE penalty is distinct from standard weight decay when they are both applied to an autoencoder with a single hidden layer of rectified linear units (ReLus).

[FRANÇAIS] En présumant l'équivalence ci-haut, décrivez si ou comment la pénalité contractante diffère du *weight decay* conventionnel lorsqu'appliquée à un auto-encodeur avec une couche cachée dont la fonction d'activation utilisée est le *rectified linear* (ReLus).

6. **(15 point) VAE.**

(a) Consider you have a latent variable model over observations $\boldsymbol{x}$ parametrized by the conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{z})$ and prior over the latent variables $\boldsymbol{z}$: $p(\boldsymbol{z})$. Imagine we want to approximate the intractable posterior distribution $p(\boldsymbol{z} \mid \boldsymbol{x})$ with a variational approximation $q(\boldsymbol{z} \mid \boldsymbol{x})$. Prove that the gap between the log probability (density) and the variational lower bound (ELBO) is the KL divergence between the true posterior and the approximation $q(\boldsymbol{z} \mid \boldsymbol{x})$. Specifically, prove :

$$\log p(x) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{x})}\right] - \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z} \mid \boldsymbol{x})}{q(\boldsymbol{z} \mid \boldsymbol{x})}\right] \tag{1}$$

[FRANÇAIS] Considérez que vous avez un modèle à variable latente qui modélise les observations $\boldsymbol{x}$ par une distribution conditionnel $p(\boldsymbol{x} \mid \boldsymbol{z})$ ainsi qu'un prior sur les variables latentes $\boldsymbol{z}$: $p(\boldsymbol{z})$. Imaginez que nous voulons estimer la distribution postérieure $p(\boldsymbol{z} \mid \boldsymbol{x})$ avec une approximation variationnelle. Provez que la différence entre la log probabilité (densité) et la *variational lower bound* (ELBO) est la KL divergence entre la vrai distribution postérieure et l'approximation $q(\boldsymbol{z} \mid \boldsymbol{x})$. Plus précisément, prouvez que :

$$\log p(x) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{x}, \boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{x})}\right] - \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}\left[\log \frac{p(\boldsymbol{z} \mid \boldsymbol{x})}{q(\boldsymbol{z} \mid \boldsymbol{x})}\right] \tag{2}$$

(b) As discussed in class, the traditional mean-field variational method corresponds to factorizing the variational approximation to the posterior distribution as a product of distributions : $q^{mf}(z_i) = \prod_j \mathcal{N}(z_{i,j}|m_{i,j}, \sigma^2_{i,j})$ and maximizing the lower bound directly with respect to the variational parameters $m_{i,j}$ and $\sigma^2_{i,j}$ for each example separately (i.e. no encoder network is learned). Could the Inverse Auto-regressive Flow (IAF) approach to inference in the VAE outperform the mean-field method ? (Explain your answer.)

[FRANÇAIS] Comme discuté en classe, la méthode *mean-field* correspond à factoriser l'approximation variationnelle de la distribution postérieure comme étant un produit de distribution : $q^{mf}(z_i) = \prod_j \mathcal{N}(z_{i,j}|m_{i,j}, \sigma^2_{i,j})$. La borne inférieure est ainsi maximiser par rapport aux paramètres variationnelles : $m_{i,j}$ et $\sigma^2_{i,j}$ pour chacun des exemples (il n'y a donc aucun encodeur !). Est-ce qu'utiliser l'approche *Inverse Auto-regressive Flow (IAF)* pour inférer nos valeurs nous permettrait d'avoir de meilleurs résultats que la méthode *mean-field* ? Expliquer votre réponse.

7. **(15 points) GANs**
For input $\boldsymbol{x} \in \mathbb{R}^N$ Consider we have a linear regression model as a very simple GAN discriminator :

$$D(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b}. \tag{3}$$

(a) Would it be advisable to use such a simple discriminator ? Justify your answer by describing what problem / benefits you might expect to encounter.

(b) For the loss used in the WGAN-GP seen in class :

$$L = \mathbb{E}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2] \tag{4}$$

where $\hat{\boldsymbol{x}}$ is interpolated between the generated $\tilde{\boldsymbol{x}}$ and a true data sample $\boldsymbol{x}$. For this simple GAN discriminator above, please provide the discriminator gradient update for the model parameters $\boldsymbol{w}$: $\nabla_{\boldsymbol{w}} L$.

(c) How does PacGAN help avoid the mode collapse problem that can plague standard GAN training ?

[FRANÇAIS] Pour $\boldsymbol{x} \in \mathbb{R}^N$, considérez que nous avons un modèle de régression linéaire comme discriminateur pour un GAN :

$$D(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{b}. \tag{5}$$

(a) Est-ce une bonne idée d'avoir un modèle aussi simple comme discriminateur ? Justifiez votre réponse en décrivant les problèmes/avantages que vous pourriez rencontrer.

(b) Voici la fonction de coût du WGAN-GP que nous avons vue en classe :

$$L = \mathbb{E}_{\tilde{\boldsymbol{x}} \sim \mathbb{P}_g}[D(\tilde{\boldsymbol{x}})] - \mathbb{E}_{\boldsymbol{x} \sim \mathbb{P}_r}[D(\boldsymbol{x})] + \lambda \mathbb{E}_{\hat{\boldsymbol{x}} \sim \mathbb{P}_{\hat{\boldsymbol{x}}}}[(\|\nabla_{\hat{\boldsymbol{x}}} D(\hat{\boldsymbol{x}})\|_2 - 1)^2] \tag{6}$$

où $\hat{\boldsymbol{x}}$ est une interpolation entre un exemple généré $\tilde{\boldsymbol{x}}$ et un vrai exemple $\boldsymbol{x}$. Pour le discriminateur linéaire décrit plus haut, donnez le gradient par rapport aux paramètres $\boldsymbol{w}$: $\nabla_{\boldsymbol{w}} L$.

(c) Comment est-ce que *PacGAN* peut nous aider à éviter le problème de *mode collapse* qui arrive souvent lors de l'entraînement de GAN classique ?

8. **(15 point) Generative models**
Consider you are training a Boltzmann machine over observations $\boldsymbol{x} \in \{0,1\}^N$ with two sets of latent variables $\boldsymbol{y} \in \{0,1\}^M$ and $\boldsymbol{z} \in \{0,1\}^K$. The joint probability is parametrized as follows :

$$P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right) \tag{7}$$

where

$$Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{y}} \sum_{\boldsymbol{z}} \exp\left(-E(x, y, z)\right) \tag{8}$$

and

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} W_{ijk} x_i y_j z_k \tag{9}$$

(a) Describe a block-Gibbs sampling strategy that will efficiently sample from sets of independent variables conditionally on the others. Derive and state clearly all required conditionals.

(b) Is the computation of the unnormalized marginal probability in $\boldsymbol{x}$ ($\log P(\boldsymbol{x}) + \log Z$) linear in the number of latent variables, as is the RBM ?

(c) Is it possible to define an efficient training strategy based on a Contrastive Divergence-like learning algorithm ? Explain your answer.

[FRANÇAIS] Considérez que vous entraîné une *Boltzmann machine* sur des observations $\boldsymbol{x} \in \{0,1\}^N$ avec deux ensembles de variables latentes : $\boldsymbol{y} \in \{0,1\}^M$ and $\boldsymbol{z} \in \{0,1\}^K$. La probabilité jointe est paramétrisée comme suit :

$$P(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\left(-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\right) \tag{10}$$

où

$$Z = \sum_{\boldsymbol{x}} \sum_{\boldsymbol{y}} \sum_{\boldsymbol{z}} \exp\left(-E(x, y, z)\right) \tag{11}$$

et

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = \sum_{ijk} W_{ijk} x_i y_j z_k \tag{12}$$

(a) Décrivez une stratégie d'échantillonnage de *blocs-Gibbs* qui échantillone à partir d'ensembles de variables conditionnellement indépendantes les unes des autres. Dérivez et énoncez clairement les distributions conditionnelles requises.

(b) Est-ce que le calcul de la probabilité marginale (non-normalisée) dans $\boldsymbol{x}$ ($\log P(\boldsymbol{x}) + \log Z$) est linéaire en fonction du numbre de variables latentes, comme pour les RBMs ?

(c) Est-il possible de définir une stratégie d'entraînement efficace qui s'inspire de l'algorithme *Contrastive Divergence* ? Expliquez votre réponse.