EN.520.665
HW2 Solution

Question 1

We will follow the notation in Prof. Chellappa's paper, "Two-Dimensional Discrete Gaussian Markov Random Field Models For Image Processing". Denote the first order Gaussian Markov Random Field (GRMF) with a singlke coefficient as the following:

$$y(s) = \sum_{r \in N} \theta y(s+r) + e(s) \tag{1}$$

where $N = \{(0,1),(0,-1),(1,0),(-1,0)\}$. Note that θ is a scalar as we only have one coefficient. Then we follow the notation to have

$$B(\underline{\theta})\underline{y} = \underline{e}, \tag{2}$$

where B is a block-circulant symmetric matrix. To ensure the bounded input bounded output stability condition, we have $\mu_s = (1 - 2\underline{\theta}^T \underline{\phi}_s) > 0$ where $\underline{\theta}$ is the M^2 vector with all the entries equal to θ. Then the eigenvalues are $1/\mu_s$ and the eigenvectors are $\underline{f}_s = Col[1, \lambda_i, \lambda_i^2 \underline{t}_j, \ldots, \lambda_i^{M-1} \underline{t}_j]$.

Question 2

An Markov chain Monte Carlo (MCMC) algorithm allows to simulate a probability distribution by constructing a Markov chain with the desired distribution as its stationary distribution. A Markov chain, defined over a set of sequential states, is an one-dimensional case of an MRF. Each state of the chain relates to a particular assignment of all involves random variables on which the stationary distribution is defined.The MCMC algorithm iteratively updates the Markov chain based on the transition probability from a state to another state. Eventually, the chain attains the state of equilibrium when the joint probability distribution for the current state approaches the stationary distribution. The parameters that leads to the stationary distribution are considered as the model parameters learnt for the particular training image.

Each state of a Markov chain is obtained by sampling a probability distribution. Among various sampling techniques, Metropolis algorithm and Gibbs sampler are two most well-known ones. Metropolis algorithm provides a mechanism to explore the entire configuration space by random walk. At each step, the algorithm performs a random modification to the current state to obtain a new state. The new state is either accepted or rejected with a probability computed based on energy change in the transition. The states generated by the algorithm form a Markov chain.

From the book of Markov Random Field Modeling in Image Analysis, the following conclusions on the two algorithms are made with a 128×128 image, (1) N = 50 iterations are enough for both algorithms, (2) Gibbs sampler tends to generate a realization with a lower energy than Metropolis sampler if N is fixed and (3) Metropolis sampler is faster than Gibbs sampler.

Question 3

Number of learnable weights in a convolution layer with N filters of size $K \times K \times C_{in}$ is $N \times K^2 \times C_{in}$, and there is a bias per each filter if those are included. Number of learnable weights in a fully-connected layer which maps input of size N_1 to output of size N_2 is $N_1 \times N_2$ or $(N_1 \times N_2) + N_2$, if biases are learnt too.

AlexNet:
First conv layer: $(11^2 \times 3 \times 96) + 96$
Second conv layer: $(5^2 \times 48 \times 256) + 256$
Third conv layer: $(3^2 \times 256 \times 384) + 384$
Fourth conv layer: $(3^2 \times 192 \times 384) + 384$
Fifth conv layer: $(3^2 \times 192 \times 256) + 256$
FC 1: $(6^2 \times 256 \times 4096) + 4096$
FC 2: $4096 \times 4096 + 4096$
FC 3: $4096 \times 1000 + 1000$

Total: 62378344

VGG16:
Conv layer 1 (conv3-64): $(3^2 \times 3 \times 64) + 64$
Conv layer 2 (conv3-64): $(3^2 \times 64 \times 64) + 64$
Conv layer 3 (conv3-128): $(3^2 \times 64 \times 128) + 128$
Conv layer 4 (conv3-128): $(3^2 \times 128 \times 128) + 128$
Conv layer 5 (conv3-256): $(3^2 \times 128 \times 256) + 256$
Conv layer 6 (conv3-256): $(3^2 \times 256 \times 256) + 256$
Conv layer 7 (conv1-256): $(1^2 \times 256 \times 256) + 256$
Conv layer 8 (conv3-512): $(3^2 \times 256 \times 512) + 512$
Conv layer 9 (conv3-512): $(3^2 \times 512 \times 512) + 512$
Conv layer 10 (conv1-512): $(1^2 \times 512 \times 512) + 512$
Conv layer 11 (conv3-512): $(3^2 \times 512 \times 512) + 512$
Conv layer 12 (conv3-512): $(3^2 \times 512 \times 512) + 512$
Conv layer 13 (conv1-512): $(1^2 \times 512 \times 512) + 512$
FC 1 (fc-4096): $25088 \times 4096 + 4096$
FC 2 (fc-4096): $4096 \times 4096 + 4096$
FC 3 (fc-1000): $4096 \times 1000 + 1000$

Total: 138357544

ResNet-18:
conv1: $(7^2 \times 3 \times 64) + 64$
conv2_x: $((3^2 \times 64 \times 64) + 64) \times 4$
conv3_x: $[((3^2 \times 128 \times 128) + 128) \times 3] + [(3^2 \times 64 \times 128) + 128]$
conv4_x: $[((3^2 \times 256 \times 256) + 256) \times 3] + [(3^2 \times 128 \times 256) + 256]$
conv5_x: $[((3^2 \times 512 \times 512) + 512) \times 3] + [(3^2 \times 256 \times 512) + 512]$
FC: $512 \times 1000 + 1000$

Total: 11511784

InceptionNet:

Conv_1: 124416 Inception (3a): 163696
Inception (3b): 388736
Inception (4a): 376176
Inception (4b): 338056
Inception (4c): 413336
Inception (4d): 512192
Inception (4e): 675328
Inception (5a): 1043456
Inception (5b): 1353968
FC: 1025000

Total: 6414360

Question 4

If the transfer function of the hidden units is linear, then we can show that a three-layer network is equivalent to a two-layer one. Let the input be x, the output at hidden layer be h and the output be y, and let the weight matrices between the input and hidden layer be W_1 and between hidden layer and output be W_2. $h = W_1x$ and $y = W_2h$, so $y = W_2W_1x$. This can be written as $y = W_3x$ where $W_3 = W_2W_1$. Hence, a three-layer network with linear hidden units can be reduced to a two-layer network.
XOR and n-bit parity problems are not linearly separable, so they cannot be solved using the above network.

Question 5

If the learning rate is too high, learning will not be stable, the loss will keep oscillating drastically and not decrease monotonically as one would like. It can be detected by monitoring the loss, when it fluctuates a lot.
If the learning rate is too low, learning will take place very slowly. The loss value will change very slowly and that can be used to detect if the learning rate is too slow.