Coding Question

Which type of neural network architecture does the following code describe?

```
class Model(nn.Module):
     def __init__(self):
          self.conv1 = nn.Conv2d(3,16,kernel_size=3,padding=1)
          self.pool = nn.MaxPool2d(kernel_size=2,stride=2)
          self.conv2 = nn.Conv2d(16, 32, kernel_size=3, padding=1)
          self.fc1 = nn.Linear(32 * 8 * 8, 128)
          self.dropout = nn.Dropout(0.5)
          self.fc2 = nn.Linear(128, 64)
          self.fc3 = nn.Linear(64, 10)
          self.batch_norm = nn.BatchNorm1d(64)

       def forward(self, x):
            x = self.pool(F.relu(self.conv1(x)))
            x = self.pool(F.relu(self.conv2(x)))
            x = x.view(-1, 32 * 8 * 8)
            x = F.relu(self.fc1(x))
            x = self.dropout(x)
            x = F.relu(self.batch_norm(self.fc2(x)))
            x = self.fc3(x)
            return x
```

A. Recurrent Neural Network (RNN)
B. Convolutional Neural Network (CNN)
C. Transformer Network
D. Long short-form memory model (LSTM)

Coding Question

What is the error in this sequential model implementation?

```
model = nn.Sequential(
    nn.Conv2d(3, 16, kernel_size=3, padding=1),
    nn.ReLU(),
    nn.MaxPool2d(2),
    nn.Conv2d(16, 32, kernel_size=3),
    nn.ReLU(),
    nn.MaxPool2d(2),
    nn.Linear(32 * 6 * 6, 128),
    nn.ReLU(),
    nn.Linear(128, 10)
)
```

A. The Conv2d layers have incorrect kernel sizes
B. The ReLU activations should be used before the Conv2d layers
C. There aren't enough layers for a deep neural network
D. It is missing a flattening operation before the linear layers

Coding Question

What is the issue with the loss reporting in this training function?

```
def train(model, train_loader, criterion, optimizer, num_epochs=5):
    # Set model to training mode
    model.train()

    for epoch in range(num_epochs):
        running_loss = 0.0

        for i, data in enumerate(train_loader):
            # Get inputs and labels
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)

            # Zero parameter gradients
            optimizer.zero_grad()

            # Forward pass
            outputs = model(inputs)
            loss = criterion(outputs, labels)

            # Backward pass and optimize
            loss.backward()
            optimizer.step()

            # Update statistics
            running_loss += loss.item()

        # Print epoch statistics
        print(f'Epoch {epoch+1}/{num_epochs}, Loss: {running_loss}')
```

A. The model is never set to evaluation mode between epochs
B. The `optimizer.zero_grad()` is in the wrong place
C. The `running_loss` is not normalized by the number of batches in `train_loader`
D. The loss calculation is using the wrong criterion

Linear Models

What is the commonly used loss function for regression problems to measure quality of prediction for a data sample (for simplicity assume that you have a linear model)?

A. Hinge loss
B. Quadratic loss (mean squared error)
C. Logistic loss
D. Cross-entropy loss

Linear Models

Which task is Not a classification problem?

A. Predicting the species of a flower based on petal measurements (rose, tulip, lily, iris, orchid).
B. Predicting whether a customer will buy a product (yes/no).
C. Predicting whether an email is spam or not based on email content.
D. Predicting the price of a used car based on its age, mileage, and brand.
E. Predicting the weather condition tomorrow (sunny, rainy, cloudy, snowy).

Linear Models

Suppose you are minimizing an empirical risk with Gradient Descent optimization. Which of the following statements is not true:

A. Using a large step-size throughout training is recommended since it guarantees convergence very fast with a very small empirical risk
B. Gradient descent starts with initializing model parameters before computing gradients
C. (Full-batch) gradient descent computes gradients over all training samples
D. Using a small step-size throughout training leads to slow convergence
E. Gradient descent updates model parameters in the opposite direction of the gradient to minimize empirical risk

Introduction to Neural Networks

For binary classification and one-hot labels, what is the cross-entropy loss of a single data point (x_i,y_i)?

A. The absolute difference between the prediction of model and ground truth
B. The probability of predicting the ground truth
C. Applying ReLU on top of a linear mapping (weighted sum of input and weights)
D. Negative logarithm of the predicted probability of the ground-truth class

Introduction to Neural Networks

Select two statements about Neural Networks that are false:

A. At each layer, the input to the activation function is the weighted sum of the outputs of the previous layers using weights and biases
B. An activation function should be always linear
C. An activation function should be differentiable to enable gradient-based optimization
D. (Logistic) Sigmoid and ReLU are examples of activation functions
E. For multi-class classification, tanh (hyperbolic tangent) activation function is used in the final layer (Layer L) so the final outputs can be interpreted as predicted probabilities
F. Logistic regression for binary classification is a neural network with one output layer and 0 hidden layers

*Hint $tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$ with output range in $(-1,1)$,
$sigmoid(x)=\frac{e^x}{1+e^x}$ with output range in $(0,1)$,
$ReLU(x)=max(0, x)$ with output range in $[0,\infty)$

Introduction to Neural Networks

Which option specifies the correct “trainable parameters” (all such parameters) that are adjusted while training a neural network with an optimization algorithm:

A. Weights and biases of all layers
B. Activation function (its type) of the final layer
C. Weights of only the first layer
D. Biases of only the final layer

Back Propagation

Consider a simple linear model $z=w\cdot x+b$, activation $a=s(z)$, and the quadratic loss $L=(a−y)^2$:

*All variables are scalars
*w,b denote the weight and bias
*$s(z)$ is an activation function and $s’(z)$ denotes the derivative of the activation function with respect to its input

What is the correct expression for $\frac{\partial L}{\partial w}$ (partial derivative of loss with respect to weight w)?

A. $2(a-y)\cdot x$
B. $(a-y)\cdot x$
C. $(a-y)\cdot s(z)$
D. $2(a-y)\cdot s’(z)$
E. $(a-y)\cdot s(z)\cdot x$
F. $2(a-y)\cdot s’(z)\cdot x$

Optimization

To minimize the empirical risk, we may use stochastic gradients to control computation complexity when the number of training data is very large. But then the variance and randomness in gradient updates may be substantial. How can one control the variance?

A. Using a larger step-size
B. Using a smaller step-size
C. Using a development set and cross-validation
D. Computing gradients over a mini batch

CNNs

Suppose you have a 2-d convolution layer with 16 input channels, 32 output channels, filter size of 3 (3x3 filter), applied to an input with spatial size of 32x32 with padding of 1 (standard). What is the total number of learnable parameters in this layer? *Do not consider biases

*Hint only the filter size and number of input and output channels matter

A. 512
B. 521
C. 9
D. 4608
E. 9216

CNNs

Select two statements about standard Convolutional Neural Networks (Encoders) that are correct:

A. Padding is the number of input elements/pixels which filter moves in every step
B. CNNs apply multiple dense (fully connected) layers in the first layers before having any convolutional layer
C. As we go deeper, the spatial size increases and the number of channels decreases
D. CNNs learn patterns by stacking simple functions in depth and many layers than learning complex functions in one layer
E. Convolution is a series (sliding window) of inner products (filter and window-view of input tensor)

CNNs

Which statement is **False** about pooling layers in CNNs?

A. Pooling reduces spatial size of activation maps
B. Pooling has a few parameters to learn that can be updated through backpropagation
C. Two common pooling methods are max pooling and average pooling
D. On activations maps with multiple channels, pooling is applied over each channel independently

Deep Architecture Evolution

Which common technique in deep learning **normalizes activations of a layer to have a zero mean and standard deviation one over elements of a mini-batch and then scales (with parameter γ) and shifts (with parameter β) the normalized outputs?**

A. Dropout
B. Residual connections
C. Pooling
D. Batch normalization
E. Fine tuning

Deep Architecture Evolution

Suppose you want to efficiently adjust the dimensionality (number of channels) without changing spatial resolution of the feature map (same height and width), e.g., in applying residual connections when the number of channels changes. What is the most common technique to adjust the dimensionality?

A. Using 1x1 convolutions
B. Using fully-connected layers
C. Max pooling
D. Average pooling
E. Using standard padding

Deep Architecture Evolution

What is NOT a key idea in the Inception/GooLeNet architecture?

A. In the forward pass, if a convolution block is not useful, it will be bypassed
B. Auxiliary classifiers at training time to encourage discrimination in the lower layers
C. Visual information processed at various scales and then concatenated along depth
D. Convolution layers in parallel with different effective filter sizes

Distribution Shifts

Suppose you have a binary classification problem with input images. Train and test images have different resolutions and are taken under different lighting conditions (this implies that conditioned on each class, train and test images have different lighting conditions and resolutions). In addition, in the train data, you have more images of one class while in the test data you have more images of the other class. What is the type of distribution shift you are facing in this example?

A. Covariate Shift
B. Label Shift
C. Class-conditional Shift
D. Full distribution Shift

Distribution Shifts

What is the main challenge to handle distribution shifts with importance weighting (IWERM)?

A. Optimization problem is non-convex
B. Lack of theoretical guarantees
C. Estimating density ratio accurately especially in high-dimensional spaces like images
D. Importance weighting does not address test data
E. Lack of labeled training data

Distribution Shifts

Suppose train and test probabilities for three classes in a classification problem under label shift are as shown in this table?

Class	$p^{te}(y)$	$p^{tr}(y)$
1	0.1	0.6
2	0.6	0.2
3	0.3	0.2

What are importance weights $r(1), r(2),\ r(3)$ for three classes?

A. $r(1) = 0.17, r(2) = 3,\ r(3) = 1.5$
B. $r(1) = 6, r(2) = 0.33,\ r(3) = 0.67$
C. $r(1) = 0.17, r(2) = 0.33,\ r(3) = 1$
D. $r(1) = 0.6, r(2) = 0.33,\ r(3) = 0.67$
E. $r(1) = 0.17, r(2) = 0.33,\ r(3) = 1.5$
F. $r(1) = 6, r(2) = 0.33,\ r(3) = 1.5$

Data Augmentation

What is the general goal of data augmentation in deep learning? Choose two options:

- **A)** Artificially expand the training data.
- **B)** Improve the models memorization of the data.
- **C)** Improve robustness to natural variations in the data.
- **D)** Make the model invariant to any distributional shifts.
- E) Make optimization faster

Data Augmentation

Why is it important to carefully choose which augmentations to apply to your training data? Choose two options.

A. Applying augmentations without careful consideration can provide data leakage, since they can add specific context which reveals the labels of the image.
B. Certain augmentations such as flipping may disrupt label consistency or violate task-specific assumptions.
C. Good augmentations depend on the specific structure and invariances of the problem domain.
D. Applying fewer, but stronger augmentations will always improve generalization.

Data Augmentation

Below is an implementation of two mystery augmentations: (Choose four options)

```
def t1(imgs: Tensor, t: float = 0.5):
    # imgs in [0,1], of shape (B,3,H,W)
    out = imgs.clone()
    mask = imgs > t
    out[mask] = 1.0 - imgs[mask]
    return out

def t2(imgs: Tensor, p: int = 5):
    # imgs in [0,1], of shape (B,3,H,W)
    out = imgs
    if torch.rand(1).item() > .5:
        s = 2*p + 1
        w = torch.zeros(3,3,s,s)
        for i in range(3):
            w[i,i] = imgs.new_ones(s,s) / s**2
        out = torch.nn.functional.pad(imgs, 4*(p,))
        out = torch.conv2d(out, w)
    return out
```

Read the code for the transforms carefully, and choose all that apply.

A) `t1` is a photometric transformation.
B) `t2` is a Gaussian blur.
C) `t2`'s parameter `p` represents a kernel radius.
D) `t1`'s parameter `t` represents an angle for rotation.
E) `t2` is deterministic.
F) `t1` applies a solarization filter.
G) `t2` applies the same transform equally to each RGB channel.

Vision Transformers

Why may we prefer to use Vision Transformers instead of CNNs?

a. Transformers are more computationally efficient

b. Transformers have fewer parameters

c. Transformers capture global context more easily

d. Transformers need less data to train

Vision Transformers

What is true about the self-attention mechanism? (Choose three options)

a. Self-attention layers are permutation equivariant
b. It is a sum of weighted value vectors
c. It replaces the need for fully connected layers
d. It reduces overfitting
e. It has a quadratic computational complexity with respect to the number of patches
f. In multi-head self-attention, every head shares the same weights

Vision Transformers

What does Query, Key and Value refer to in self-attention?

a. Fixed vectors computed from the network input
b. Linear projections of the same input
c. Trainable weight matrices
d. Vectors that describe how the model should weigh a layer’s output

Image Segmentation

Given an input vector of shape (13x13) a kernel of shape (3x3) and stride=1, the result of a **transposed** convolution has output shape (9x9). What was the padding parameter of this operation?

A. 0
B. 1
C. 2
D. 3
E. 4
F. 5

Image Segmentation

What is true about semantic segmentation?

a. Only care about foreground objects
b. Detects individual objects
c. Classifies all pixels in an image
d. Draws bounding boxes around objects and fills the boxes with correct class label

Image Segmentation

What is different about dilated convolution, compared to standard convolution?

a. It is the opposite operation
b. It inserts holes between kernel elements
c. It reduces the number of needed filters
d. It compresses the receptive field to reduce computational cost

Recurrent Neural Networks (RNNs)

Which of the following statements are true about gradient clipping-by-norm?

A. It rescales the gradient vector whenever its norm exceeds a threshold.
B. It sets gradients above a threshold to the threshold value.
C. It guarantees the prevention of the vanishing gradients problem.
D. It is important to use gradient clipping-by-norm when using the sigmoid activation function.

Recurrent Neural Networks (RNNs)

Which input-output structure of a Recurrent Neural Network (RNN) would you use to classify sentences into two categories: positive, negative?

A. One-to-one
B. One-to-many
C. Many-to-one
D. Many-to-many
E. Many-to-many (encoder-decoder style)

Recurrent Neural Networks (RNNs)

Which of the following are true about Truncated Backpropagation Through Time (Truncated BPTT)? (Choose two options)

A. It trades off training speed for inference speed.
B. It trades off capturing long-range dependencies for reduced memory requirements.
C. Truncated BPTT cannot be used to train an RNN with LSTM cells.
D. Truncated BPTT requires unrolling the RNN across time steps.

Adversarial Attacks

Which of the following statements best describes the main challenge posed by adversarial examples for deep learning models?

A) They are created by random perturbations, for which the models are already robust.
B) They reveal that small, carefully chosen changes to inputs can consistently fool models without affecting human perception.
C) They affect models generalization, particularly models trained with limited data.
D) They show that black-box models, for which we don’t fully understand the model and its parameters, cannot be trusted.

Adversarial Attacks

What makes designing adversarial attacks in black-box settings challenging? Choose two options:

A. It only allows for universal, untargeted attacks.
B. The attacker must rely on querying the model and observing only input-output behavior, which can require many queries to find a successful attack.
C. Transferability of adversarial examples between different models does not always work.
D. They are limited to stochastic perturbations, since black-box models cannot use gradient methods.
E. You can only use the fast gradient sign method to make it work

Adversarial Attacks

Given a target neural network, we generate a single perturbation by optimizing (with access to gradients) over a batch of images, pushing the model's predictions away from the original class for each image. This perturbation is then added to new, unseen images.

Which of these options does not describe the attack?

A. White-box attack
B. Per-sample attack
C. Untargeted attack

Object detection

Which of the following are the two stages in a two-stage detector (e.g., Faster R-CNN)?
(Choose two options)

A. Generate candidates using a Region Proposal Network (RPN)
B. Use Non-Maximum Suppression to filter out overlapping candidates
C. Slide a window over the raw image to assign a class to candidates
D. Classification and bounding-box regression for the candidates generated

Object detection

To handle the imbalance of background vs. foreground predictions, one can use: (Choose two options)

A. Feature Pyramid Network (FPN) to generate predictions
B. Hard Negative Mining (selecting only the difficult background predictions)
C. Focal Loss (down-scaling the loss for well-classified background predictions)
D. Bipartite matching between predictions and ground truths

Evaluation Methods in Deep Learning for Image Analysis

Which of the following statements are true? (**Select TWO**)

A. Cross-validation estimates are often overly influenced by one training subset.
B. Spurious correlations are a known issue in deep learning for medical imaging, where models may learn to use unintended visual features rather than actual disease-related patterns.
C. Internal datasets are often considered representative of the real-world setting because the data are assumed to be independent and identically distributed (iid), which is typically valid in practice.
D. Evaluating multiple models on an external test set and selecting the best helps reduce bias and should be done for robust evaluation.
E. Data augmentation can help mitigate domain shift when visual factors such as lighting, background, or device differences affect model generalization.

Performance Evaluation Metrics

Which of the following statements are true? (**Select TWO**)

A. AUROC can be interpreted as the probability that a randomly chosen positive receives a higher score than a randomly chosen negative.
B. Each point on the Precision-Recall curve reflects a specific False Positive Rate (FPR) and True Positive Rate (TPR) at a specific decision threshold.
C. Scaling prediction scores by a constant (e.g., dividing by 10) does not affect the AUC, but it changes Average Precision metrics.
D. A model with 60% sensitivity and 90% specificity has the same balanced accuracy as one with 90% sensitivity and 60% specificity.
E. A model improves its Top-1 accuracy from 70% to 80%, but its Top-5 accuracy remains at 90%. We can say that the false negative rate remains unchanged.
F. In multiclass classification with macro-averaging, precision, recall, F1-score, and accuracy are always equal. (Assume each sample has a single correct label.)

Macro-Averaged Precision

A deep learning model classifies images into three categories: **Labrador Retriever**, **Golden Retriever**, and **Border Collie**. The model produces the following performance on a test set:

Class	**F1-score**	**True Positives (TP)**	**False negative (FN)**
Labrador Retriever	0.72	36	12
Golden Retriever	0.72	42	14
Border Collie	0.80	40	10

Assume:

- One label per sample (single-label classification)
- Macro-averaged precision is defined as the mean of class-wise precisions

Hint:

The F1-score is the harmonic mean of precision and recall:

F1 = (2* Precision* Recall) /(Precision + Recall)

What is the model's macro-averaged precision?

A. 0.77
B. 0.75
C. 0.73
D. 0.71