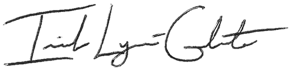
**CSCI 5922 Exam**

**Honor Statement:**

**I, Isaiah Lyons-Galante, understand that I will do this exam alone and without the help of other humans. I understand exactly what it means to “do my own work” and I will do my own work. If I fail to do this and/or my work looks similar to other work, I understand that very bad things can happen.**

****

**Signature \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Date December 5, 2023**

**Directions: Please complete this Exam using THIS DOCUMENT.**

**Download and save this document.** Place all answers, work, illustrations, images, etc. that you want graded into this document.

If you are asked to code, you can create you code in Jupyter, as .py, on git, or whatever. You just need to have a link to your code that you will place on this document. Therefore, when you submit this exam, you will submit only this document.

**Please save this document as: YourName\_NN\_Exam\_2023.docx.**

**Notes and Rules:**

1. **No questions are permitted** by anyone for any reason during this exam. Follow the instructions, do not overcomplicate things, and make assumptions (that you clearly write down) if/as needed. Part of the test is your ability to “do the test”.
2. **It is not permitted to work with any other humans** on this Exam. While I am not concerned about this as I feel that all of you are very ethical, I am required to note that anyone who works together must get a “0” grade and may both fail the class and potentially have further issues with the program. Just do your own work 😊
3. **This Exam is open** – meaning you can use the web, class notes, my website, my code, your code, etc. **If you are in doubt about something – do not use it**. For example, if your buddy Bob posts code on the web that answers one of these questions and then you use it – that’s cheating. Using my code or your code is fine.
4. **This Exam will be due no later than 12/10 (Sunday) by 11:59 pm MT**. If you wait until the last hour to submit and run into a problem, there will be no solution. This Exam cannot be late for any reason. Please do not test this. Submit EARLY – like 24 hours early! I have set up the submission area so that you can **submit as many times as you want.** So, you can submit in advance (like Saturday), then you can do more work on Sunday if you wish, and then you can submit again if you want. **We will use the LAST (latest) submission** for grading. This way, there is no need to wait until the last minute to submit. Again, to be clear, the **system will close and will lock at 11:59pm MT on 12/10. Exams not submitted by that time will not be graded.** Also – for those of you who always push things – if you submit the “wrong version” or a “poem for your honey” instead of the Exam – that will **not matter!** **BE CAREFUL**. **Submit early and submit correctly.** (It makes me sad that I have to say all of this 😐)

**Part 1: Common Interview Questions** and Learning New Things. This part of the Exam will ask you to define new terms and/or answer questions. Please keep your answers brief, succinct, concise, and precise, and never more than **5 sentences**. (more than 5 sentences can result in points lost)

Think of this as an interview.

1. What is an activation function (in neural networks) and why is it used?

*An activation function is a mathematical operation that is applied to the output of a neuron in a neural network to generate an “activated” output. Because neurons are typically a linear sum-product of weights and input values, they have difficulty approximating non-linear functions, such as the infamous XOR problem. Activation functions are typically non-linear, such as sigmoids or hyberbolic tangents, which give the network the ability to approximate non-linear functions.*

1. What is an exploding gradient and give an example of what could cause it.

*An exploding gradient is a quickly increasing derivative of the loss function with respect to the model parameters. The gradient is used to determine the direction and size of the change of model parameters when performing backpropagation, in tandem with the learning rate. If the gradient is increasing too much, the model will not converge to a minimum. This could occur easily in a recurrent neural network that has long series of inputs because the gradient for some parameters is a function of all prior inputs. If the values in this chain are > 1, their product and thus the gradient can get very large very quickly.*

1. When using an activation function in a CNN that predicts images, why might you choose the ReLU?

*ReLU’s power is in the simplicity of its calculation and of its derivative—for values below 0, the derivative is 0, and for values above 0, the derivate is 1. Furthermore, any negative values are simply reduced to a 0. This is important in a CNN designed to work with images because the input typically has really high dimensionality and thus the model has many layers with many parameters. ReLU helps to keep the derivatives and thus the gradient within reasonable values.*

1. A transformer (such as for language translation) has an encoder and a decoder. Suppose you have a word that is one-hot encoded. What 4 things will a common encoder include when embedding that word? Hint: The first one is some kind of embedding. What are the other three?

*A transformer encoder commonly includes first the embedding of the word into a vector space that captures the meaning of the word. Next, positional encoding is added to capture the location of the word in the input. Next, a self-attention mechanism captures the relative importance of the word in relation to the other words in the input. Finally, normalization and feed forward layers encode all of the above, typically into other encoder blocks.*

1. Define BERT (Bidirectional Encoder Representations from Transformers).

*BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained natural language processing model developed by Google. It utilizes a transformer architecture and is trained on vast amounts of text data to learn contextualized representations of words. BERT is bidirectional, meaning it considers both left and right context in a sentence, enabling it to capture richer semantic meanings. It has achieved state-of-the-art results in various NLP tasks by fine-tuning its pre-trained weights on specific downstream tasks.*

1. What is cross attention in a transformer?

*Cross-attention is a mechanism by which a transformer model pays varying degrees of attention to different parts of the input sequence when generating the output sequence. Unlike self-attention, which considers relationships within a single sequence, cross-attention involves attending to positions in the input sequence based on the context provided by another sequence.*

1. What is transfer learning?

*Transfer learning is a machine learning model training technique that involves pre-training the model on a different but similar dataset to the target task before fine-tuning on the specific task at hand. It is a powerful technique that allows a model to learn on a vast trove of labeled or unlabeled data before attempting a task that might have much less data.*

1. Define GAN (Generative adversarial network)?

*A GAN is a type of neural network model that learns from the competing aims of a generator and a discrimator. The discriminator is attempting to learn to distinguish between a real input and a fake one, while the generator simultaneously learns to create fake inputs that are indistinguishable from real ones. They achieved impressive results in the field of generative AI.*

1. Define GPT.

*GPT stands for Generative Pre-Trained Transformer. It refers to the family of language models based on transformers that are pre-trained on massive amounts of diverse text data in order to learn contextualized representations of words. The generative part refers to the fact that after these models are prompted with input, they generate a bunch of coherent and contextually relevant text.*

1. Define ChatGPT.

*ChatGPT is a software product created by OpenAI in late 2022 that brought a lot of joy to students and a lot of headaches for teachers accustomed to assigning boring, rote assignments. Why? Well, based on the GPT defined above, OpenAI made it possible for anyone with an internet connection to prompt their large language model GPT3.5 and get a coherent response. ChatGPT has since passed the bar exam, med school exams, and written likely millions of high school English essays across the world. If ChatGPT can answer this question, does that make it self aware?*

1. Common activation functions include ReLU, sigmoid, tanh, and softmax (among others). Give an example when you would use the sigmoid as the last activation function in a NN. Give an example when you would use softmax as the last activation function in a NN.

*Sigmoid is a very useful activation function that is excellent for binary classification. Because it’s shape, it normalizes the input value to between 0 and 1 and for most real numbers, the value is very close to 1 or 0. The softmax function is similarly useful for classification, but instead for multiclass problems. It takes a whole range of input weights and normalizes them such that they are all between 0 and 1 and add up to one, making the outputs interpretable as probabilities that the input belongs to a given class.*

1. Suppose you have labeled input data where the labels are one-hot encoded. Suppose also that your labels can be one of three categories (like dog, cat, mouse for example). Next, suppose the last activation function of your NN is the softmax. Which Loss function would you choose to use in this case and why?

*For this case, I would most certainly use categorical cross entropy because this is a classification problem with more than 2 classes. CCE rewards a model not just for guessing the correct category, but it also rewards guessing correctly with high confidence. It therefore performs excellently at training the model progressively, helping the model get slowly more and more confident as it trains.*

1. Why use max pooling CNNs – what does max pooling do?

*Max pooling is a type of layer whose primary purpose is to reduce the dimensionality of the features in a model. It does this by defining a pool, or patch, of a given size (say 3 by 3) and pulling out the maximum value from within that pool. This simplifies the input (in our case 9 values) down to just a single value that still captures a lot of the important signal. It’s commonly used with images which have notoriously high dimensionality to get us from millions of pixels down to feature space of just a few thousand categories.*

**Part 2: Architectures - Derivatives - and Keras**

For this part of the exam, you will illustrate your understanding of the basic/common architectures and in simple cases the derivatives for back propagation. This part of the exam will ask you to “draw” architectures. You may do this using the “draw” option in Word, by using “shapes” in Word, or if you must, by hand (and then insert your drawing). You may NOT copy/paste from the web. Why? Because I want YOU to create. Here you will also be asked to “fully label” your networks. What this means is that you will note in a smart way where all the weights are, where the biases are, what the activation functions are, and what the loss function is. You will also properly note/label hidden layers and units, or for the CNNs the filters and kernels, etc. In other words, all the parts of the networks should be designated. Finally, you will be asked to show the derivative(s). When asked this, read what you are being asked to do and then meet that requirement. **The best thing to do is to show all your work and as much as you can.** Remember, not showing something can mean one of two things. It can mean that its just too obvious to show or it can mean that you do not know it and so did not show it. Unfortunately, the graders must assume the latter. So, be safe, and show your work.

**Question 1**

1. Create and fully label an ANN that has an input for 4D data and has two hidden layers. The first hidden layer will have 4 units, the second hidden layer will have three units, and the output will have three units. For the first hidden layer, use the Sigmoid. For the second hidden layer use the ReLU, and for the output layer use the softmax. Use the categorical cross entropy as your Loss function.
2. Choose any weight between your first and second hidden layers and circle it – so we know the one you selected. Next, write out the derivatives in the chain that you would need to calculate the update for that weight. Once you write out the chain of derivatives, then derive the gradient equation. When you do this – show your work. What are the values of each derivative? Note – make any assumption you feel you need. (Do not overcomplicate this. You are just writing out the chain rule of derivatives and then showing their values.) **Notes: You are not using any numbers here, so your “answers” will not be numbers.**
3. Create and **paste here** the Keras code that would create this exact network. I prefer that you use the Keras API sequential model which starts with:

**My\_NN\_Model = tf.keras.models.Sequential([ ….**

However, there are many ways to show/do this in Keras and you may choose how to show your model.

Include the model definition, the model summary, and the model compile. You do not need to fit or run the model. For “compile” you choose what you want in there.

Part 2 Question 1A: Draw the ANN

Diagram

Description automatically generated

Part 2 Question 1B: Derivative of the Loss Function with Respect Weight w211

**#### Equations for ANN:**

Z1 = X \* W1 + B

H1 = σ(Z1)

Z2 = H1 \* W2 + C

H2 = ReLU(Z2)

Z3 = H2 \* W3 + D

ŷ = softmax(Z3)

L = CCE(y, ŷ)

L = -y \* log(ŷ)

**#### Derivative ∂L/∂W2\_11:**

∂L/∂W2\_11 = ∂L/∂ŷ \* ∂ŷ/∂Z3 \* ∂Z3/∂ZH2 \* ∂H2/∂Z2 \* ∂Z2/∂W2\_11

∂L/∂ŷ = -y/ŷ

∂ŷ/∂Z3 = ŷ \* (1 - ŷ)

∂Z3/∂H2 = W3

∂H2/∂Z2 = 1 if Z2 > 0 else 0

∂Z2/∂W2\_11 = H1

if Z2 > 0:

∂L/∂W2\_11 = -y/ŷ \* ŷ \* (1 - ŷ) \* W3 \* 1 \* H1

∂L/∂W2\_11 = -y \* (1 - ŷ) \* W3 \* H1

if Z2 <= 0:

∂L/∂W2\_11 = -y/ŷ \* ŷ \* (1 - ŷ) \* W3 \* 0 \* H1

∂L/∂W2\_11 = 0

Part 2 Question 1C: Create the ANN in Keras

# import tensorflow

import tensorflow as tf

# create the model

model = tf.keras.models.Sequential([

tf.keras.layers.Dense(4, activation='sigmoid', input\_shape=(4,)),

tf.keras.layers.Dense(3, activation='relu'),

tf.keras.layers.Dense(3, activation='softmax')

])

# compile the model

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

# print the model summary

model.summary()

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

dense\_6 (Dense) (None, 4) 20

dense\_7 (Dense) (None, 3) 15

dense\_8 (Dense) (None, 3) 12

=================================================================

Total params: 47

Trainable params: 47

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Question 2**

Create a Neural Network that takes in a single greyscale image that is 30 by 30. The network will have two filters, each that use a different 3 by 3 kernel to filter the image using the Keras Conv2D default for the stride and the padding as “same”. The next step in the network will perform 2 by 2 max pooling. The next step will use 4 filters (each with its own 3 by 3 kernel – stride (1,1) – and padding = “same”). The next step will be 2 by 2 max pooling. The next step will flatten and send into a fully connected NN. The output will have three units (three outputs are generated) and will use softmax. All other activation functions (except the last which is softmax) will be ReLU.

1. Draw this architecture. Make sure it is fully labeled with all sizes and shapes as well as other critical elements.
2. Create this architecture in Keras and paste the code here. Specifically, you will need to paste the code your used to define the model, to summarize the model, and to compile the model. In the compile portion, you can use “adam”, “categorical\_crossentropy”, and “accuracy” for your metrics. You will not include fitting or running the model. Please also paste the results of your model summary here.

Part 2 Question 2A: Draw the CNN

**Application, arrow

Description automatically generated**

Part 2 Question 2B: Write the CNN in Keras

# import tensorflow

import tensorflow as tf

# create the model

cnn = tf.keras.models.Sequential([

tf.keras.layers.Conv2D(filters=2, kernel\_size=(3, 3), activation='relu', padding='same', input\_shape=(30, 30, 1)),

tf.keras.layers.MaxPooling2D(pool\_size=(2, 2)),

tf.keras.layers.Conv2D(filters=4, kernel\_size=(3, 3), activation='relu', padding='same'),

tf.keras.layers.MaxPooling2D(pool\_size=(2, 2)),

tf.keras.layers.Flatten(),

tf.keras.layers.Dense(units=64, activation='relu'),

tf.keras.layers.Dense(units=3, activation='softmax')

])

# compile the model

cnn.compile(

loss = 'categorical\_crossentropy',

optimizer = 'adam',

metrics = ['accuracy']

)

# print the model summary

cnn.summary()

Model: "sequential\_3"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 30, 30, 2) 20

max\_pooling2d (MaxPooling2D (None, 15, 15, 2) 0

)

conv2d\_1 (Conv2D) (None, 15, 15, 4) 76

max\_pooling2d\_1 (MaxPooling (None, 7, 7, 4) 0

2D)

flatten (Flatten) (None, 196) 0

dense\_9 (Dense) (None, 64) 12608

dense\_10 (Dense) (None, 3) 195

=================================================================

Total params: 12,899

Trainable params: 12,899

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Part 3: Applying Neural Nets (ANN, CNN, LSTM) to real labeled text data.**

For this part of the Exam, I have gathered articles on three topics: football, science, and politics. The data has already been cleaned, tokenized, and vectorized. Each row (vector) in the dataset is an article, each row is also labeled as foo tball, science, or politics. Each column is a word in the vocabulary. The data itself represents the number of times each word appear in that given article. (The data was gathered from newsapi.org) .

**Here is a link to the cleaned, prepared, labeled data.**

<https://drive.google.com/file/d/1-ZAbxWN29iCo44kaLSfYmV2E8YDKcgGE/view?usp=sharing>

---

(If you want to know how this was done (**not required**) – here is code and a tutorial) <https://gatesboltonanalytics.com/?page_id=254>

---

**The overall goals here include:**

1. Coding, comparing, and using an ANN, CNN, and LSTM RNN in TF/Keras (Python) to Train models and to Test their accuracy.
2. You want to see if you can predict the topic of an article (in this case – football, science, or politics).
3. You also want to compare and illustrate the accuracy of your models and determine/discuss which model (ANN, CNN, or LSTM) is best and why this might be.
4. **It is up to you how to do this and how best to illustrate and explain your steps, results, and conclusions. Assume the reader is non-technical.**
5. You will include a **link** to your code, but do **not paste or otherwise include code on this Exam** document. (Again, you can place your code wherever you want as long as there is a link to it)

**Specific Requirements:**

There are many ways to do this. The following offers a few core requirements. Beyond this, **YOU must decide what to do and how best to do it**. Part of your grade will be based on your flow, discussion, illustrations, report, and communication of methods and results. Again, you will post a link to the code, but you will not include or paste code here.

1. Use Python and TF/Keras to Train and then Test the accuracy for an ANN, CNN, and LSTM RNN. In other words, you will use three different Neural Networks to create models that should predict whether a test vector (which represents an article on a topic) is on the topic of *science*, *football*, or *politics*. You will need to write code to do this. You already have code for ANNs, CNNs, and LSTM RNNs, so you may choose to repurpose/update your code as needed.
2. To show your work and to **illustrate and explain** your work, results, and conclusions you must include at least the following:
3. A link to your code. If you wish, you can put your code on your website, Google Colab, GitHub or wherever, and then include the URL here.
4. Show and explain how you **prepared the data** so that you can use it properly to Train and Test your models. (You are not required to validate – but you certainly can). Specifically, if you split the data, discuss and illustrate this. If you encode the labels, discuss and illustrate this, etc. Use images (like screenshots) as needed. YOU decide and explain/show what you are doing.
5. **DO NOT** include or paste any code. You do not need nor should you use “code” to explain or illustrate what you are doing. Use illustrations, images, explanations. Pretend that the person grading this paper does not know Python but does want to see and understand what you did, what you found, how your models compare, which model worked best, etc.
6. TO be clear - You will be coding, training, and then testing three types of models: ANN, CNN, LSTM. Therefore, you should include screen images (small portions) of the training for each (a few of the last epochs), as well as **confusion matrices** for each that illustrate the test data accuracy for each model.
7. Discuss and describe what you are doing and showing.
8. Discuss and illustrate the results. Which model worked best (have confusion matrices that support this discussion). Comment on which model you expected to work the best, which model actually worked the best and why.

**From here, you determine what is needed.**

You are welcome to use my code to assist (or other examples from the web or the Keras website)

**Introduction**

In this part of the exam, we aim to leverage three different types of Neural Networks - Artificial Neural Network (ANN), Convolutional Neural Network (CNN), and Long Short-Term Memory Recurrent Neural Network (LSTM RNN) - to predict the topics of articles on football, science, and politics. The data has been gathered from newsapi.org, cleaned, tokenized, and vectorized, with each row representing an article and each column representing a word in the vocabulary.

**Code Repository:**

You can find the code implementation for this task on my website at <https://isaiahlg.com/csci5922/final/final.html> or on GitHub at <https://github.com/isaiahlg/csci5922/blob/main/final/final.ipynb>.

**Data Preparation:**

The data started in a format that looked like the screenshot below. In order to prepare it to be modeled, a few key steps were taken.

A screenshot of a computer

Description automatically generated with medium confidence

Data Shuffling: the data was sorted by label, and so the first step was the shuffle the data.

A screen shot of a computer

Description automatically generated with low confidence

Data Normalization: the values represented the number of times the word appeared, and so they needed to be normalized to range between 0 and 1. To do this, we find the minimum and maximum values of x, and then apply the formula xscaled = ( x – xmin ) / (xmax – xmin).

Label Encoding: The labels ('football', 'science', 'politics') were one-hot encoded for the model to understand and learn from them for classification. After encoding, the y-values looked like this:

Graphical user interface, application

Description automatically generated

Data Splitting: The next step was to divide the full dataset into training and testing sets to evaluate the models accurately. A common split ratio of 80/20 train/test was used. The new shapes of the data were:

Text

Description automatically generated

**Creating the Models**

For consistency of evaluation across model types, all three models were designed to have between 9,000 – 12,000 parameters. This is right about 10X the number of training samples we have, a good rule of thumb in deep learning.

1. **ANN:** the simplest of the three models, the ANN has three fully connected layers with 30, 10, and 3 units respectively. Dropout is used between layers with a dropout rate of 0.2 to avoid overfitting. The activation function for the 2 hidden layers is sigmoid because this is a classification problem, and softmax for the output layer since there are multiple categories. We use the adam optimizer and categorical cross entropy loss.
2. **CNN:** though CNNs are typically for 2D images, 1D CNNs can be used for single dimensional input such as this text data. We use two convolutional layers with 5 and 15 filters respectively. Both use ReLU as activation, and are followed by a 2x2 max pooling layer. Then we flatten the output, and put it through 2 dense layers with 0.3 dropout, finishing with a softmax activation. We use the adam optimizer and categorical cross entropy loss.
3. **LSTM:** LSTMs are typically applied to sequential data, however, this data is not sequential, so our expectations are low. We add two LSTM layers of 40 and 20 units respectively, followed by 0.2 dropout and a single dense layer with softmax activation. The model struggles to converge, and so we set the learning rate manually to 0.001, using categorical cross entropy as well.

**Model Training and Performance**

For consistency of evaluation across model types, all three models were run for 80 epochs and with a batch size of 100. This batch size was found to be a good compromise between slow speed with small batches, and unstable training with large batches. The number of epochs was found to be the amount of time after which the training accuracy tended to surpass the peak validation accuracy, and the model would begin to overfit, reducing the validation accuracy.

1. **ANN:** The ANN ran the most efficient and performed the best of the three models. The overall accuracy was 74.2%. This impressive accuracy was achieved in just 4.5 seconds of training. Many different sizes and depths were tried, and yet this surprisingly simple network was able to achieve performance comparable to models with 20X more parameters. Find the training accuracy by epoch below as well as the confusion matrix for the final prediction on the test data.

Chart

Description automatically generated Chart, treemap chart

Description automatically generated

1. **CNN:** The CNN was a close second place to the ANN, achieving an accuracy of 71.9%. This means that the CNN convolution layers were successfully able to extract meaningful features from the word vectors using convolutional filters. The model was able to train in a relatively efficient 12.7 seconds. Find the training accuracy by epoch below as well as the confusion matrix for the final prediction on the test data.

**Chart

Description automatically generated Chart, treemap chart

Description automatically generated**

1. **LSTM:** The LSTM performed much worse than the first two models, achieving an accuracy of 51.5%. At first, it wouldn’t converge at all, with the accuracy remaining in the 30%s, as good as random guessing. Finally, manually setting the learning rate enabled it to learn something from the data, though not much. This tell us that the nature of the input data is barely sequential. This makes sense since the word vectors appear alphabetically, not in the order in which they appear in the article. This hampers the ability of the LSTM to learn from sequences of data. In addition, despite having almost the same number of parameters as the CNN, the LSTM took a glacial 5 minutes and 15 seconds to train. Find the training accuracy by epoch below as well as the confusion matrix for the final prediction on the test data.

**Chart, line chart

Description automatically generated** **Chart

Description automatically generated**

**Conclusion**: The ANN peformed the best on this dataset, and also happens to be the smallest, and most efficient of the models. On top of that, experiments show that the accuracy of the ANN remained above 70% even when scaled down to just 3000 parameters (10, 5, and 3 units). I had originally expected the LSTM to work best because text data is typically sequential. However, for this type of tokenized data, an ANN is certainly the way to go.