Analysis of Trainable Parameters in Artificial Neural Networks

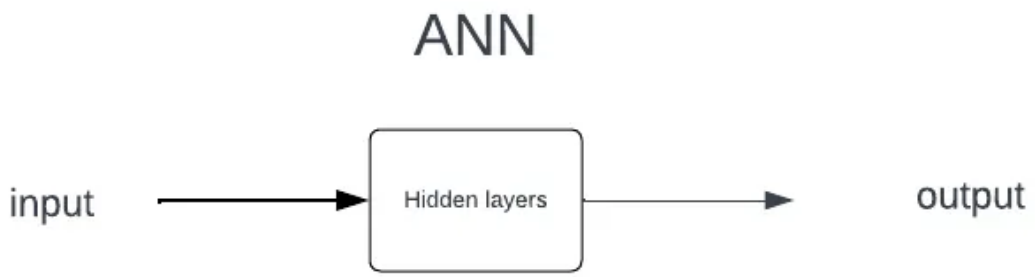
5 min read · Sep 20, 2023

R Raushan Agrawal Follow

Listen Share

We know that Weights and Biases are the Heart of Neural Networks.

When you hear about Artificial Neural Networks (ANNs), you might picture them as complex black boxes that perform magic. In reality, they are powerful mathematical models that can be broken down into simpler components. Two crucial components are “weights” and “biases.” In this blog post, we’ll dive into the essential concepts of calculating trainable parameters (weights and biases) in ANN hidden layers, breaking it down into simple steps.



“The Structure of an ANN”

what is node?

A node, also known as a neuron, is like a tiny decision-making unit. It takes in information, weighs its importance, and decides whether to activate or not, based

on a certain rule. These nodes work together to process data and make predictions, making them the basic building blocks of the network's intelligence.

How many number of input node in Ann's?

Input nodes neurons corresponds to the number of features in your dataset. Each input node represents one feature or variable from your dataset. Therefore, if you have a dataset with, for example, 3 different features (such as age, income, education level), you would typically have 3 input nodes in the input layer of your ANN.

How many number of hidden layers?

The number of hidden layers depends on the problem's complexity, and it's a crucial aspect of designing an effective neural network for a specific task. Experimentation and tuning are often required to find the optimal architecture.

How many number of output layers?

The output layer is the final layer that produces the network's predictions or classifications. The number of neurons (nodes) in the output layer depends on the problem type:

1.For regression tasks (predicting continuous values), there is typically one neuron in the output layer.

2.For binary classification (yes/no or 1/0), there is one neuron.

3.For multi-class classification (categorizing into multiple classes), there is one neuron per class, each representing the probability or confidence score of belonging to that class.

"Understanding Weights and Biases"

What Are Weights and Biases?

Weights: Think of weights as the strength of connections between neurons in an ANN. Each connection has a weight, which determines how important the input from one neuron is to the output of another. These weights are adjusted during training to make the network learn and generalize better.

Biases: Biases are like an offset or a threshold for each neuron. They allow neurons to activate even when the weighted sum of inputs is not enough. Biases help ANNs account for different starting points, ensuring they can learn complex patterns.

“Calculating Trainable Parameters”

How many Number of weights and bias (Trainable Parameter)?

The number of trainable parameters (weights and biases) in an Artificial Neural Network (ANN) depends on its architecture, specifically the number of neurons in each layer, including the input, hidden, and output layers. Here’s a formula to calculate the total number of trainable parameters in an Artificial neural network:

Total Trainable Parameters = (Number of Neurons in Input Layer * Number of Neurons in Hidden Layer 1) + (Number of Neurons in Hidden Layer1 * Number of Neurons in Hidden Layer2) + …+(Number of Neurons in Last Hidden Layer * Number of Neurons in Output Layer) + (Number of Neurons in Hidden Layer + Number of Neurons in Output Layer)

In this formula:

- Number of Neurons in Input Layer: The number of input features or nodes.

- Number of Neurons in Hidden Layer: The number of neurons in each hidden layer.

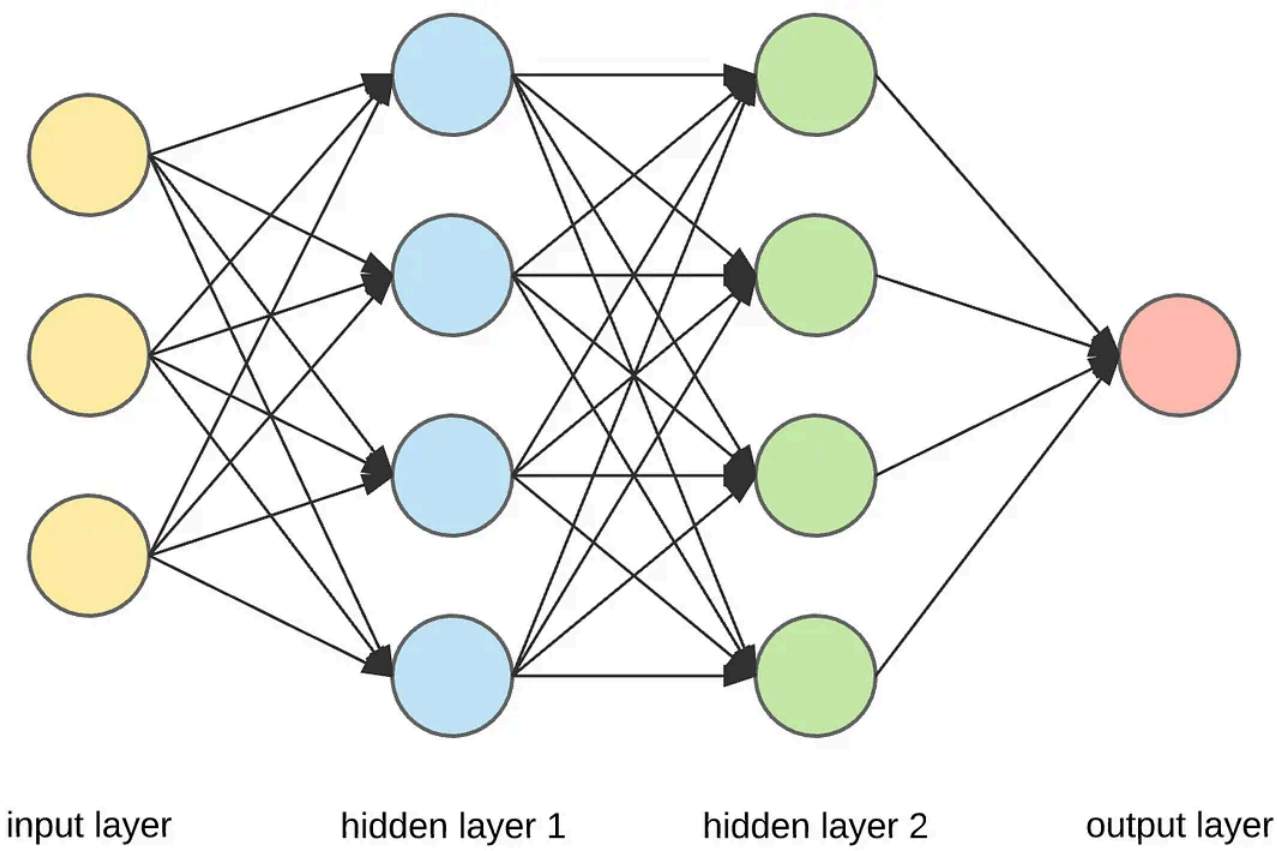
- Number of Hidden Layers: The total number of hidden layers.

- Number of Neurons in Output Layer: The number of neurons in the output layer, which depends on the problem (e.g., binary classification — 1 neuron, multi-class classification — number of classes neurons, regression — 1 neuron).

It's important to note that each weight (connection between neurons) and each bias (one for each neuron) is a trainable parameter. The number of weights and biases is summed up to determine the total trainable parameters in the network. The specific values will vary based on the architecture and problem you're working on.

Examples

Let's consider a simple Artificial neural network with the following architecture:



- Input Layer: 3 nodes

- Hidden Layer 1: 4 nodes

- Hidden Layer 2: 4 nodes

- Output Layer: 1 node

Calculating Weights:

1. Between Input Layer and Hidden Layer 1:

- Each of the 3 input nodes connects to each of the 4 hidden layer 1 nodes, resulting in 3 * 4 = 12 weights.

2. Between Hidden Layer 1 and Hidden Layer 2:

- Each of the 4 hidden layer 1 nodes connects to each of the 4 hidden layer 2 nodes, resulting in 4 * 4 = 16 weights.

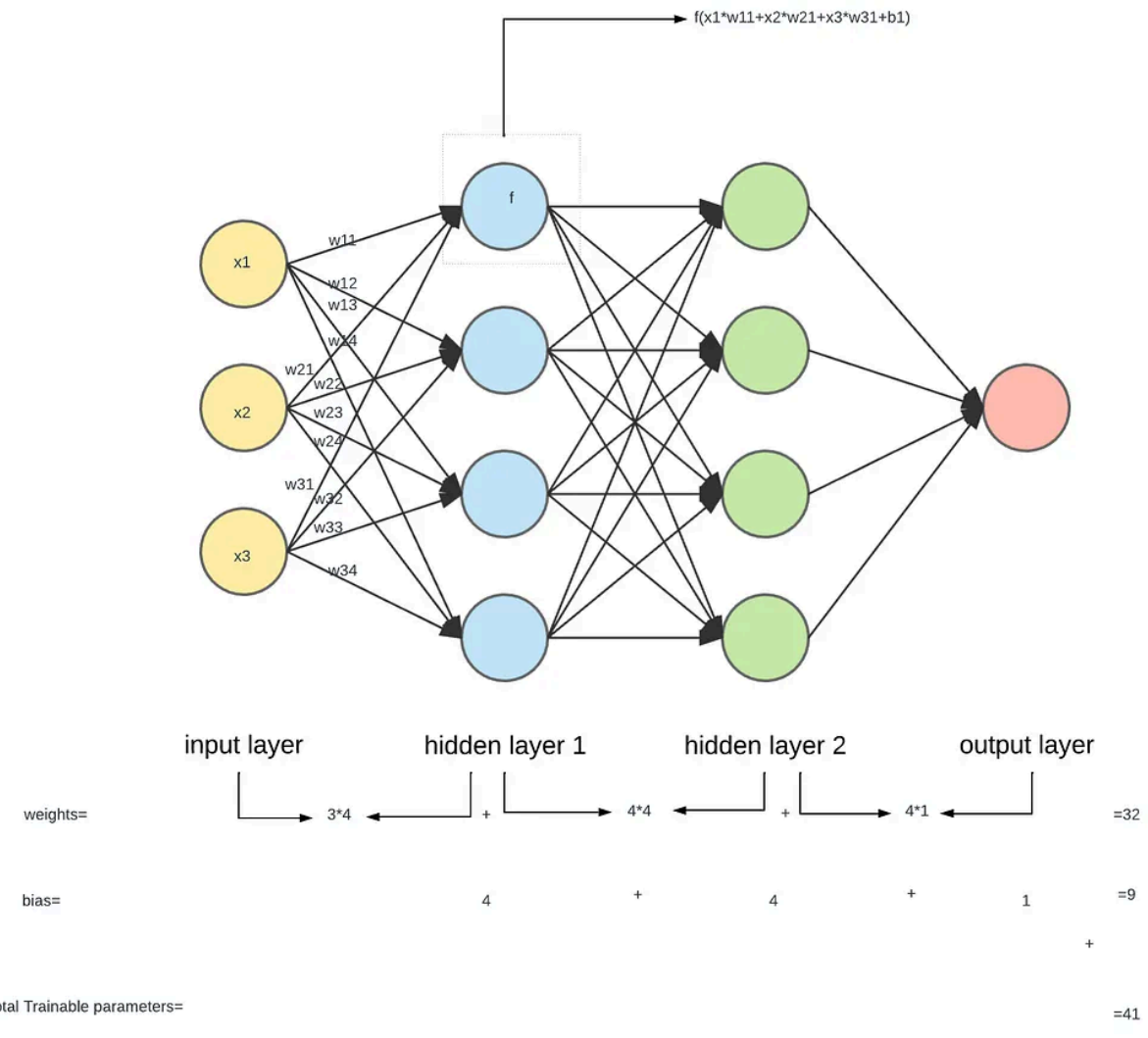
3. Between Hidden Layer 2 and Output Layer:

- Each of the 4 hidden layer 2 nodes connects to the single output layer node, resulting in 4 * 1 = 4 weights.

Calculating Biases:

- Each node in the hidden layers and output layer has its bias. So, you have 4 biases in Hidden Layer 1, 4 biases in Hidden Layer 2, and 1 bias in the Output Layer, totaling 4 + 4 + 1 = 9 biases.

Diagram:



In this example, there are a total of 12 + 16 + 4 = 32weights and 9 biases, making a total of 32 + 9 = 41 trainable parameters in the neural network.

```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

```
model = Sequential()

model.add(Dense(4,input_dim=3))
model.add(Dense(4))
model.add(Dense(1))
```

```
model.summary()

```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 4)                 16

 dense_1 (Dense)             (None, 4)                 20

 dense_2 (Dense)             (None, 1)                 5

=================================================================
Total params: 41
Trainable params: 41
Non-trainable params: 0
_________________________________________________________________
```

12weights,4bias

16weights,4bias

4weights,1bias

The exact number of weights and biases will vary depending on the number of nodes in each layer and the network's architecture. In larger and more complex networks, the number of trainable parameters can become much larger, potentially millions or even billions, which highlights the importance of efficient training algorithms and architectures.

Ann Neural Networks

Follow

Written by Raushan Agrawal

9 followers · 6 following

AI/ML Engineer|| BHUain

Responses (2)

Write a response

What are your thoughts?

Ansh kapoor
Aug 10, 2024

Can you give a generic formula from these observations

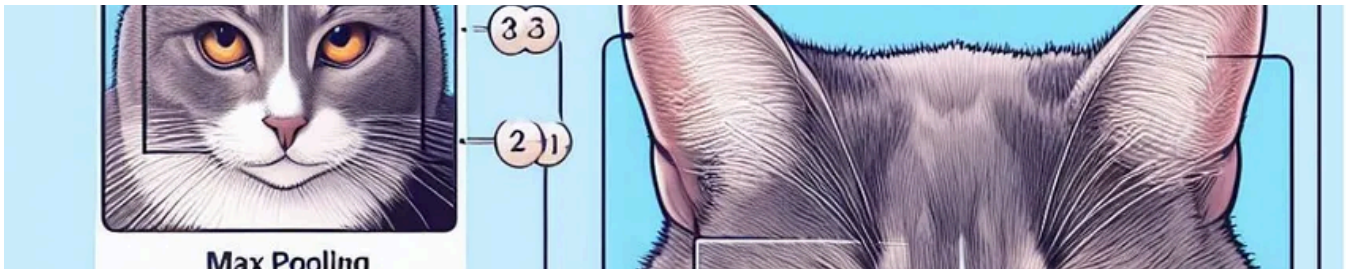
1 reply Reply

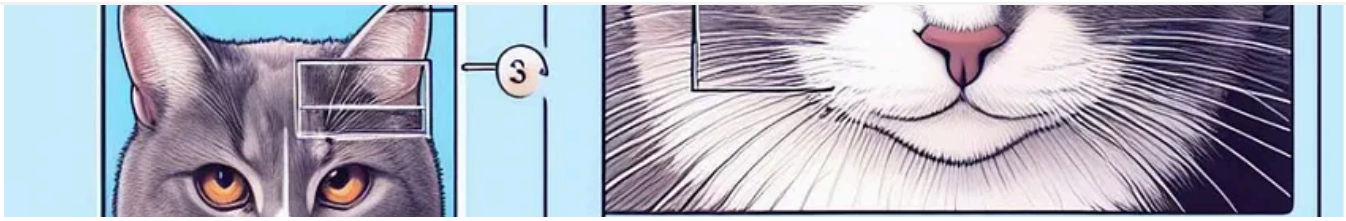
Achitktr
Sep 20, 2023

Nice explanation.

Reply

More from Raushan Agrawal





R Raushan Agrawal

A Deep Dive into the world of Convolutional Neural Networks

1. Introduction

Nov 2, 2023

See all from Raushan Agrawal

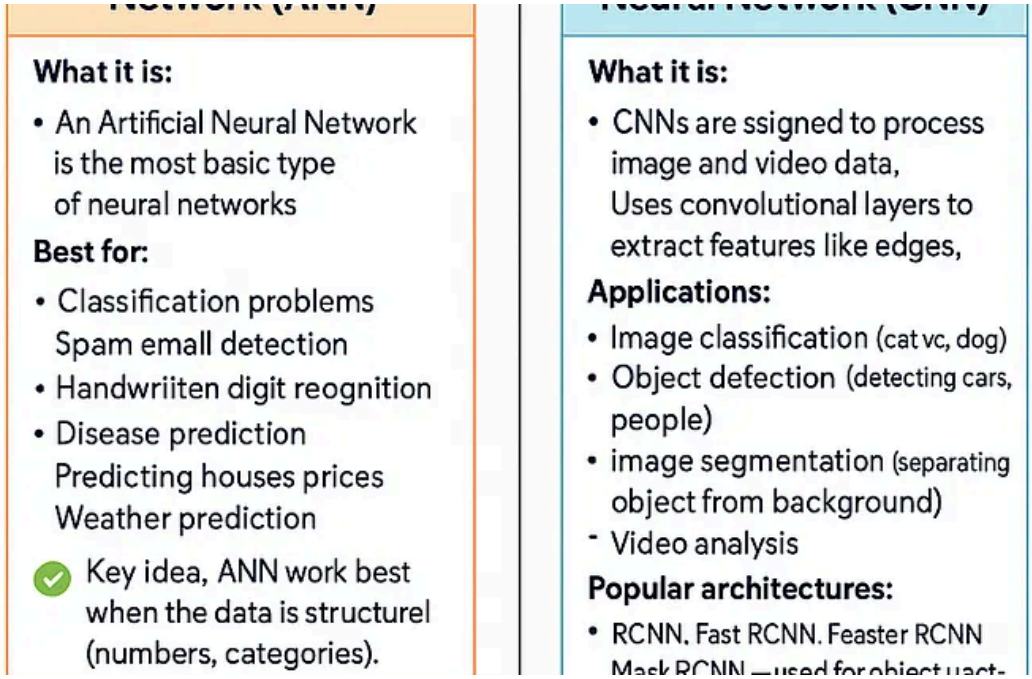
Recommended from Medium

Yashwanth S

Retrieval Augmented Generation (RAG) 06 :- BM25 Retriever: When and Why to Use It (With Code Demo)?

Retrieval is the first and most crucial step in many Natural Language Processing (NLP) applications like search engines, chatbots, and...

May 28 6

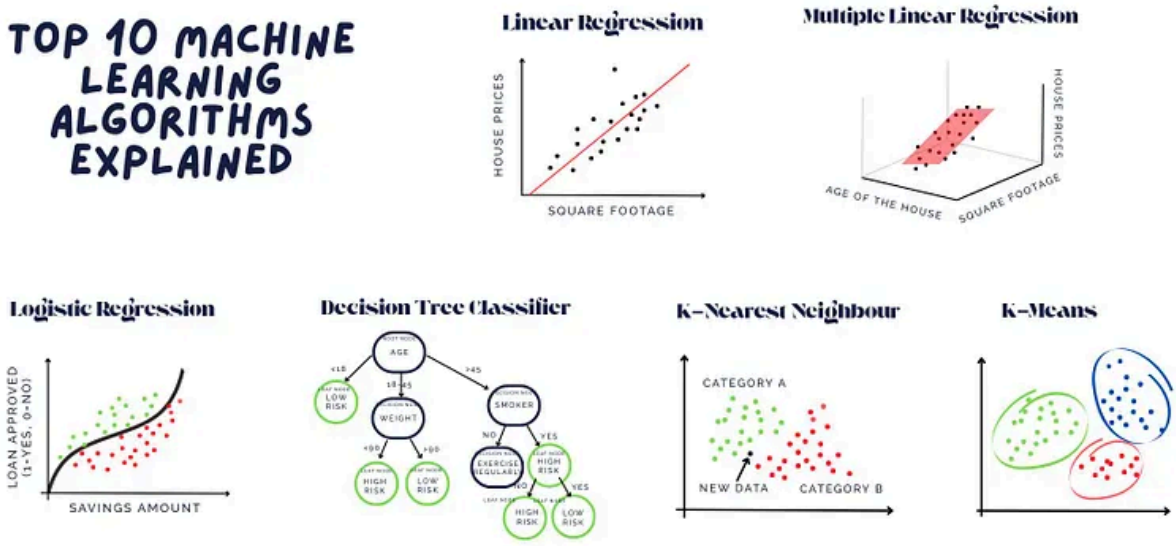


Sanjay Singh

Deep Learning Neural Networks Explained: ANN, CNN, RNN, and Transformers (Basic Understanding)

Deep Learning is at the heart of modern Artificial Intelligence. From image recognition to language translation, neural networks power...

Aug 17 5

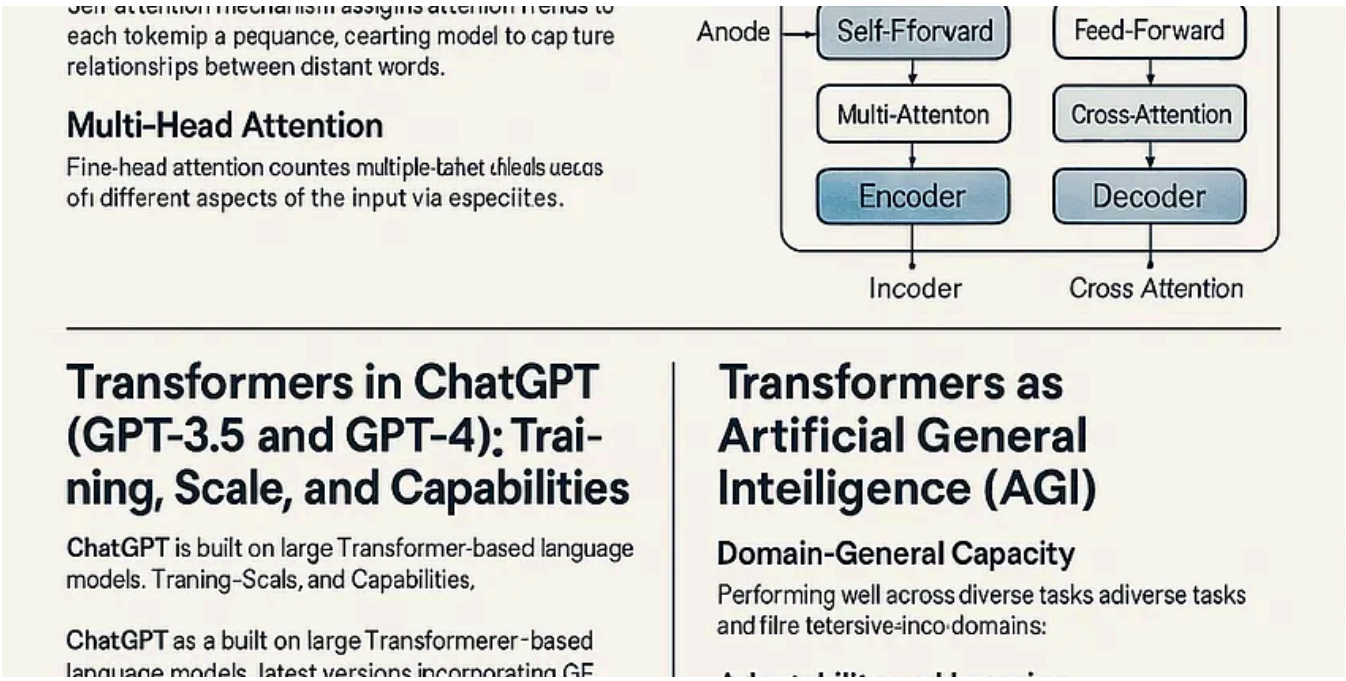


In Learning Data by Rita Angelou

10 ML Algorithms Every Data Scientist Should Know—Part 1

I understand well that machine learning might sound intimidating. But once you break down the common algorithms, you'll see they're not.

Jun 10 31



Marxism-Leninism

Transformer Neural Networks and ChatGPT: Structure, Training, and why they are AGI (Now and for...

Transformer Architecture: Structure and Key Components

Aug 9 1

MatMaq

Why XGBoost Champions Are Switching to LightGBM (And You Should Too)

The gradient boosting revolution that's quietly dominating Kaggle leaderboards and production systems worldwide

Aug 5 119 4