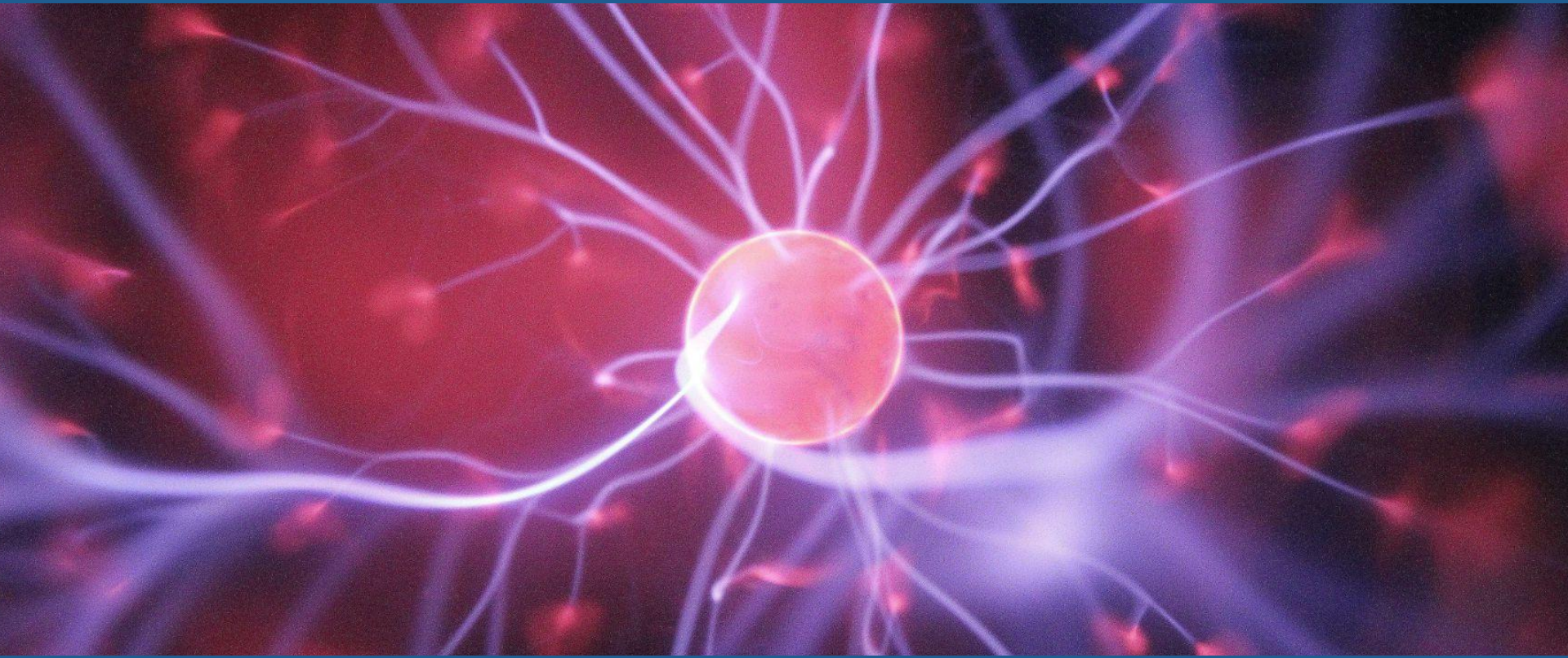
MALARIA DETECTION



ANGELA TSENG x DEEP LEARNING

Executive Summary

Problem: Malaria remains a **global health challenge**, with over 229M cases and 400,000 deaths in 2019.

Solution: Early and accurate diagnosis is vital to treatment. I developed a **deep learning solution to automate blood cell image analysis for malaria** in order to accelerate manual microscopy.

Key Findings: I evaluated base plus four model variants. **Model 3** emerged as the optimal solution, delivering **98.6% test accuracy with the highest recall for parasitized cells (99%) and a lightweight architecture.** It catches virtually all infections with minimal false negatives, a priority for patient safety.

Recommendation: **Adopt Model 3 in the malaria detection workflow** to minimize missed infections and enable fast, cost-effective deployment. Labs and clinics can dramatically speed up diagnosis, achieve higher accuracy, and mass deploy malaria screening in resource-constrained regions.

Expected Benefits: Implementing Model 3 could reduce image review from minutes to seconds, **enabling higher throughput and faster treatment decisions**. Eliminating inter-observer variability will **improve reliability, increase early malaria detections, save expert labor, and improve patient outcomes**, aligning both public health and business efficiency goals.

Problem Definition

Malaria is a contagious disease

Caused by Plasmodium parasites transmitted to humans through infected mosquitoes bites

Half of the world's population is at risk

Especially in sub-Saharan Africa. Over 229 million cases and 400,000 deaths reported in 2019

Children under 5 are the most vulnerable

Accounted for 67% of malaria deaths worldwide in 2019

Traditional diagnosis ismanual & slow

Delays and inter-observer variability lead to missed diagnoses and continued high mortality

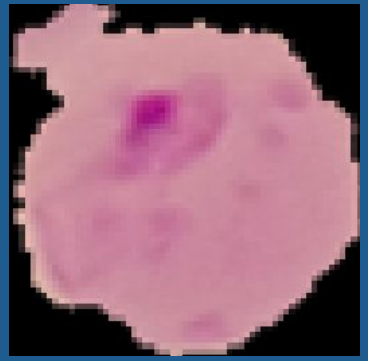
Problem to Solve

Project Goal: Build an automated malaria detection **deep learning** model to classify red blood cell images as parasitized vs uninfected and exceed expert-level accuracy and speed.

Target Outcome: A model that reduces diagnostic time, improves detection sensitivity, and flags infected cells with high confidence. Success means **fewer cases of malaria go undetected**.

Rationale: Even a small improvement in **speed or sensitivity** can translate into **millions of lives saved** and more efficient use of limited resources.

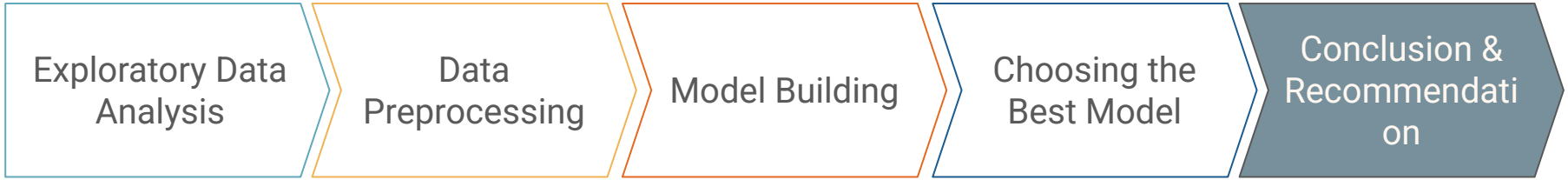
parasitized



uninfected

Dataset with **24,958 train** and **2,600 test** images

Solution Approach



- Load the data
- Data overview
- Visualize data
- Summarize key findings

- Convert RGB to HSV images
- Gaussian blurring
- Split the data
- One hot encoding

- Build base model
- Check model performance
- Tune model with 4 variations

- Define success metrics
- Compare model performance
- Choose the best model

- Final solution design & business recommendations
- Refine insights
- Understand risks and challenges

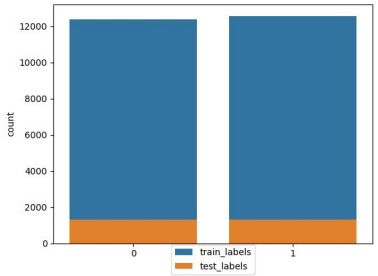
Key Findings: Exploratory Data Analysis

Examine dataset shape

24,958 train and 2,600 test images; size 64×64 pixels with 3 color channels. Labels are binary (parasitized vs uninfected)

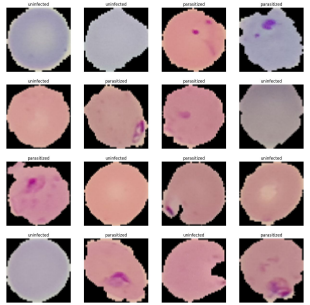
Check for class imbalance

No significant imbalance (parasitized v uninfected are roughly 50/50)



Normalize & visualize images

Pixel values scaled to 0–1 for faster, more stable model training



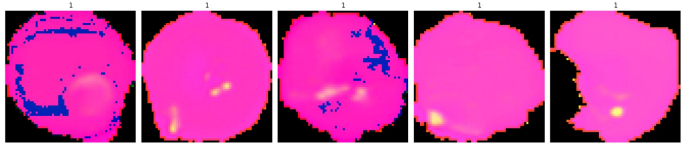
Observe visual distinctions

Shapes and colors don’t signal infection. Parasitized cells have **darker spots within**

Key Findings: Data Preprocessing

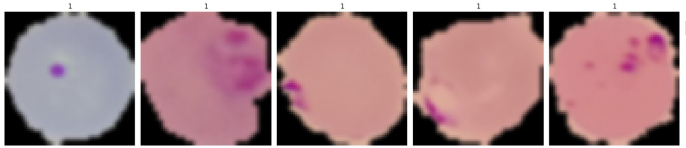
Convert RGB to HSV images

Significant **color separation and intensity** (blue & yellow spots) inside infected cells



Gaussian blurring

Reduce noise while retaining parasite blobs-no new insight



Split the data

Partition training data into train and validation sets (80/20 split)

One hot encoding

Encode the class labels for binary, 0/1 encoding

Takeaway: Visually examining the images reveals good data quality and clear distinguishing features conducive for model training

Key Findings: Model Building

Base Model	A simple CNN with 3 conv layers (each followed by max pooling) and 2 dense layers. This is my baseline performance.
Model 1	Increase depth - 4 conv layers and 3 dense layers - to explore if a deeper network improves accuracy.
Model 2	Improve activation & regularization - 3 conv layers, each with BatchNorm, LeakyRelu, max pooling, and a dense layer (BN+LR).
Model 3	Augment data with optimized architecture - 3 conv layers with more filters and kernel size (with max pooling) and a smaller dense layer.
Model 4	Transfer learning approach - extract and flatten pretrained VGG16 network and add 3 dense layers with BatchNorm to classify malaria.

Key Findings: Choosing the Best Model

Decision Criteria

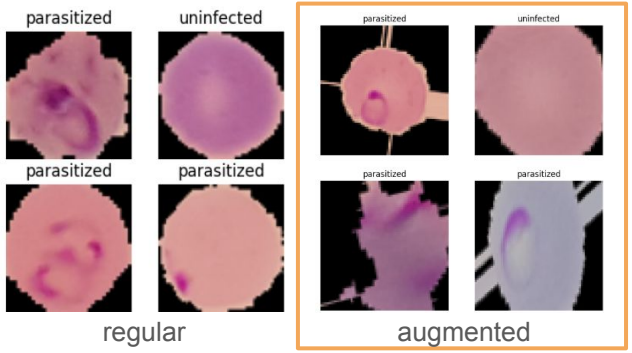
- Recall on parasitized cell is the most critical metric
- Missing an infection (false negative) is a worse outcome than a false alarm
- Model parameter counts (indicate size) affect deployment feasibility
- Test accuracy

	Parasitized (1) Precision/Recall	Total Parameters	Test Accuracy
Base	0.99/0.98	1,058,786	98.6%
Model 1	0.97/0.97	617,890	97.2%
Model 2	0.99/0.96	1,143,106	97.5%
Model 3	0.98/0.99	580,994	98.6%
Model 4	0.95/0.93	15,289,410	97.5%

Model 3: Model Selection Rationale

1. Aggressive data augmentation

- Flipping, rotating, and zooming focus model on intrinsic cell features; improve rotation- and translation- invariance
- Benefits outweigh feature loss



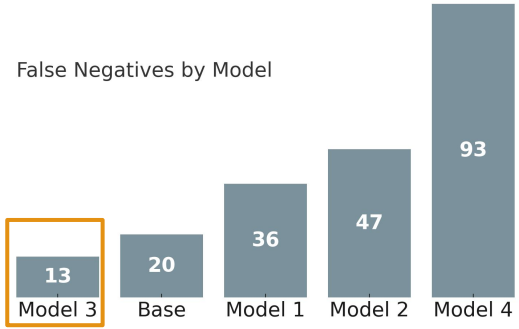
2. Compact architecture with 45% less parameters

- Increase layer filters to 64
- Larger kernel 3x3 for richer feature extraction
- Reduce dense layer to 128 neurons to avoid overfitting

```
model3 = Sequential()
model3.add(Conv2D(32, 3, activation='relu', padding='same', input_shape=(64, 64, 3)))
model3.add(MaxPooling2D(2))
model3.add(Conv2D(64, 3, activation='relu', padding='same'))
model3.add(MaxPooling2D(2))
model3.add(Conv2D(64, 3, activation='relu', padding='same'))
model3.add(MaxPooling2D(2))
model3.add(Flatten())
model3.add(Dense(128, activation='relu'))
model3.add(Dropout(0.3))
model3.add(Dense(2, activation = "softmax"))
```

3. Recall maximization for infected cells

- Model catches almost every infected cell, resulting in very few false negatives
- Missing an infection could lead to late treatment and mortality



Business Recommendation

Data Science & IT
Prepare model for deployment (set up servers or embed in device). Establish a pipeline of new blood smear images.

Management
Allocate budget and resources for necessary equipment and training. Monitor program implementation and impact.

Lab Operations
Develop new procedures to integrate AI detection with human review. Train technicians to use the new system.

Partners and Regulators
Align community partners on AI detection benefits. Engage medical device regulators to ensure policy compliance

Cost and Benefit Analysis

	COST	BENEFIT
Development & Integration	One-time cost. Integrate Model 3 into user-friendly applications for technicians, and to set up necessary infrastructure.	**Improved Speed & Scale** - Model 3 can be deployed in multiple locations on standard hardware without expensive GPU servers.
Hardware & Equipment	Fixed cost. Digital microscopy equipment and inexpensive computing devices. Can repurpose existing lab infrastructure without expansion.	**Enhanced Accuracy & Consistency** - Process bulk image volume without additional cost. Images can be reused for further analysis.
Training & Adoption	Upfront cost. Technicians will need training to learn the new system. Anticipate initial slowdown and resistance to change.	**Lab Efficiency & Labor Savings -** Remove mundane tasks and free up skilled labor for patient care and confirming false positives.
Maintenance & Updates	Ongoing cost. Model retraining needed if new data emerges - when parasite mutates, adapt to new hospital systems.	**Strategic Advantage** - Easy to retrain Model 3 for sustained accuracy and maintain leading edge of delivering high-quality medical service.
False Positive Follow up	Time for an expert to quickly double-check false alarms either with a human review or a secondary test.	**Save Lives** - Rare false positives with Model 3 and faster/cheaper to double check result than missing an infection that becomes fatal.

Implementation Risk and Challenges

- **Data Shift Risk:** Images from different hospitals, microscopes, staining techniques may differ widely to degrade performance
- **Computing Constraints:** Limited compute resources (power & device) or internet connectivity in remote areas
- **Ongoing Maintenance:** Software bug or hardware failure could disrupt diagnoses without a support or fallback process
- **User Adoption & Training:** Technicians may show reluctance to change routine if tool is not user-friendly and benefits are unclear
- **False Positives Burden:** Redesign workflow so that images flagged are efficiently reviewed and cleared with minimal effort
- **False Sense of Security:** Over-reliance on AI so technicians skip further review such as spot-checking and additional testing
- **Regulatory and Ethical Risks:** Maintain transparency, data privacy, and document results to avoid oversight and liability risks

Next Steps

Pilot Deployment: Limited-scope Model 3 deployment in a controlled setting. Run model in parallel with standard diagnostics as a proof-of-concept to uncover and address real-world issues before scaling up.

Gather Additional Data & Retraining: Expand training dataset to fine-tune Model 3 to adapt to local specificities. Plan for periodic retraining and evaluation if new data indicates changes.

Develop Deployment Infrastructure: Cloud-based API or model running on local device. Integrate software with existing lab information systems and workflow.

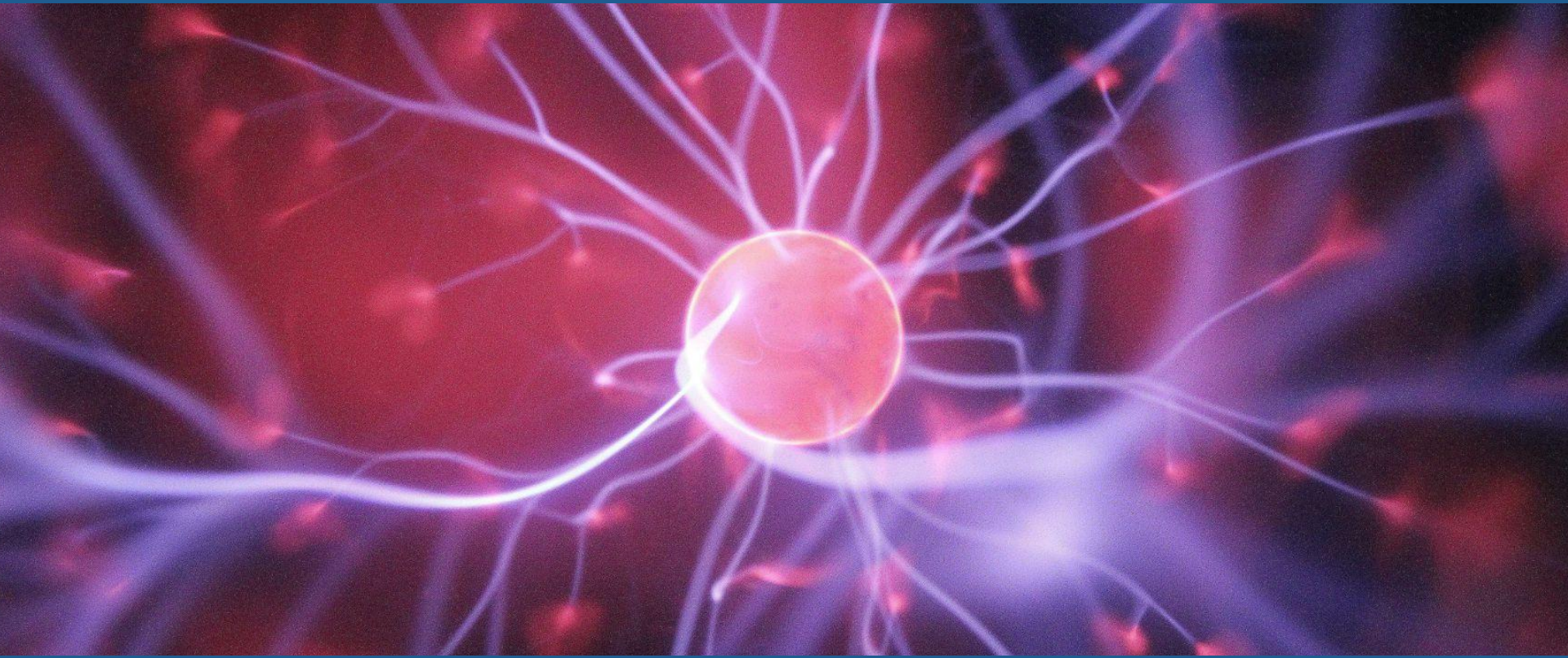
User Training & Documentation: Include a quick-start guide, result interpretation, and procedure for handling positive cases. Conduct training at pilot site to get technicians comfortable with the tool.

Establish Feedback Loop: Dashboard to track key metrics for troubleshooting and model improvement.

Iterate and Scale Up: Deploy model to other labs and regions in a phased manner.

Communicate Success: Secure ongoing support, raise team motivation, and attract partnerships.

THANK YOU

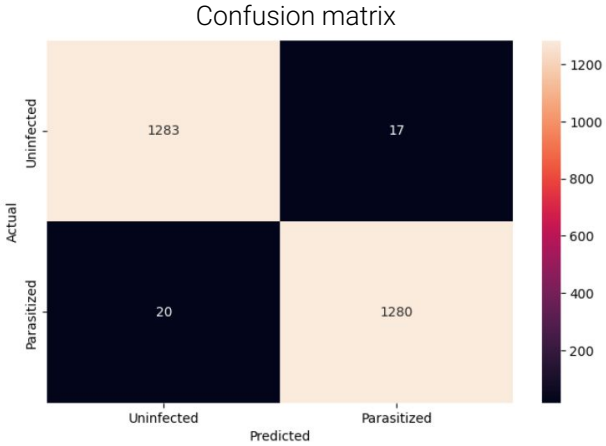


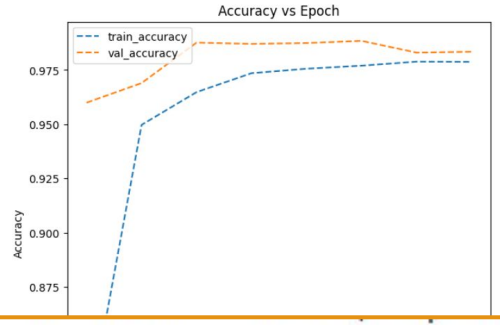
QUESTIONS?

Appendix: Base Model

Observations: 98.6% test accuracy; high recall; simple design,
some inefficiencies - low filter/small kernel/large dense layer

```
model = Sequential()
model.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu", input_shape = (64, 64, 3)))
model.add(MaxPooling2D(pool_size = 2))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size = 2))
model.add(Dropout(0.2))
model.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu"))
model.add(MaxPooling2D(pool_size = 2))
model.add(Dropout(0.2))
model.add(Flatten())
model.add(Dense(512, activation = "relu"))
model.add(Dropout(0.4))
model.add(Dense(2, activation = "softmax"))
```





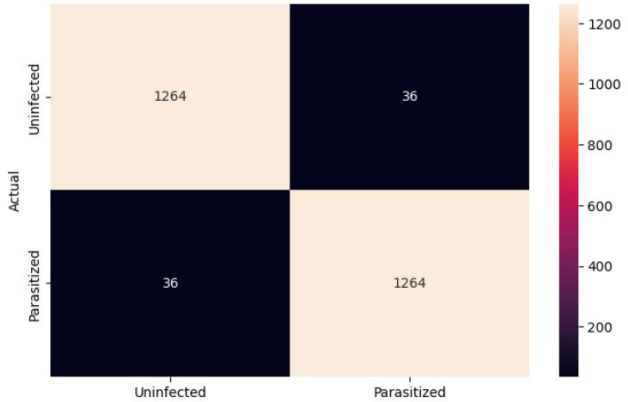
	precision	recall	f1-score
0	0.98	0.99	0.99
1	0.99	0.98	0.99

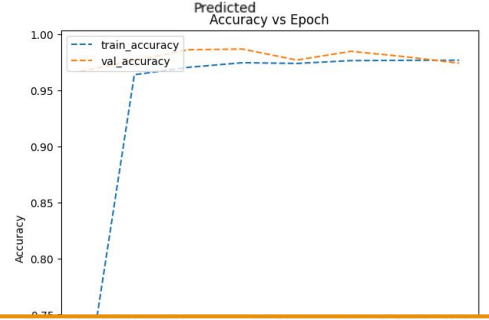
Appendix: Model 1

Observations: 97.2% test accuracy, lower recall; deeper not wider; more aligned train/val accuracy; sub optimized new layers/dropout

```
model1 = Sequential()
model1.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu", input_shape = (64, 64, 3)))
model1.add(MaxPooling2D(pool_size = 2))
model1.add(Dropout(0.2))
model1.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu"))
model1.add(MaxPooling2D(pool_size = 2))
model1.add(Dropout(0.2))
model1.add(Conv2D(filters = 32, kernel_size = 2, padding = "same", activation = "relu"))
model1.add(MaxPooling2D(pool_size = 2))
model1.add(Dropout(0.2))
model1.add(Conv2D(filters = 64, kernel_size = 3, padding = "same", activation = "relu"))
model1.add(MaxPooling2D(pool_size = 2))
model1.add(Dropout(0.3))
model1.add(Flatten())
model1.add(Dense(512, activation = "relu"))
model1.add(Dropout(0.4))
model1.add(Dense(128, activation="relu"))
model1.add(Dropout(0.3))
model1.add(Dense(2, activation = "softmax"))
```

Confusion matrix





	precision	recall	f1-score
0	0.97	0.97	0.97
1	0.97	0.97	0.97

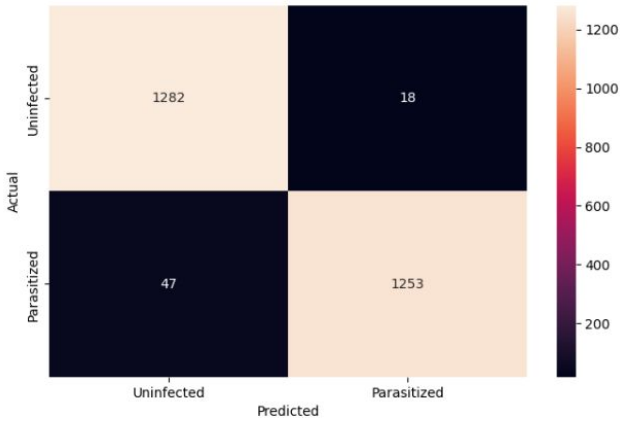
Appendix: Model 2

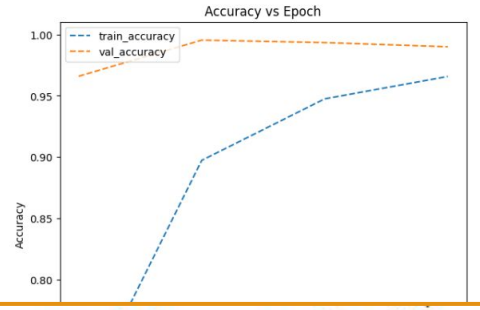
Observations: 97.5% test accuracy, lower class I recall; high Dropout and BatchNorm seem to over regularize

```
model2 = Sequential()

model2.add(Conv2D(32, (3, 3), input_shape = (64, 64, 3), padding = 'same'))

model2.add(BatchNormalization())

model2.add(LeakyReLU())

model2.add(MaxPooling2D(pool_size=2))

model2.add(Dropout(0.2))

model2.add(Conv2D(64, kernel_size = 3, padding = "same"))

model2.add(BatchNormalization())

model2.add(LeakyReLU())

model2.add(MaxPooling2D(2))

model2.add(Dropout(0.3))

model2.add(Conv2D(128, kernel_size = 3, padding = "same"))

model2.add(BatchNormalization())
```

```
model2.add(Conv2D(128, kernel_size = 3, padding = "same"))

model2.add(BatchNormalization())

model2.add(LeakyReLU())

model2.add(MaxPooling2D(2))

model2.add(Dropout(0.4))

model2.add(Flatten())

model2.add(Dense(128, activation = "relu"))

model1.add(BatchNormalization())

model2.add(LeakyReLU())

model2.add(Dropout(0.3))

model2.add(Dense(2, activation = "softmax"))

adam = optimizers.Adam(learning_rate = 0.0005)
```

Confusion matrix





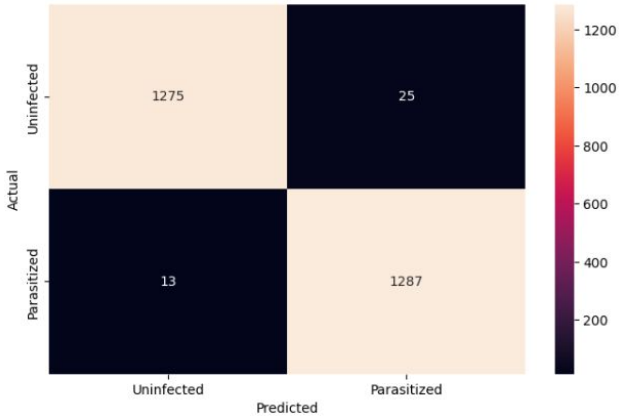
	precision	recall	f1-score
0	0.96	0.99	0.98
1	0.99	0.96	0.97

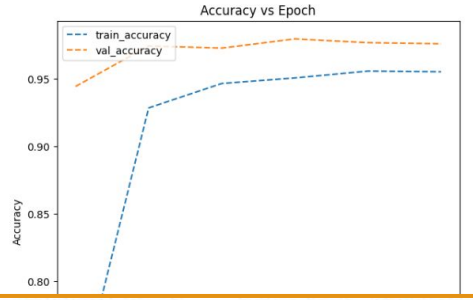
Appendix: Model 3

Observations: 98.6% test accuracy, highest Class I recall. 45% less parameters than base model. Efficient and elegant architecture

```
model3 = Sequential()

model3.add(Conv2D(32, 3, activation='relu', padding='same', input_shape=(64, 64, 3)))

model3.add(MaxPooling2D(2))

model3.add(Conv2D(64, 3, activation='relu', padding='same'))

model3.add(MaxPooling2D(2))

model3.add(Conv2D(64, 3, activation='relu', padding='same'))

model3.add(MaxPooling2D(2))

model3.add(Flatten())

model3.add(Dense(128, activation='relu'))

model3.add(Dropout(0.3))

model3.add(Dense(2, activation = "softmax"))
```

Confusion matrix





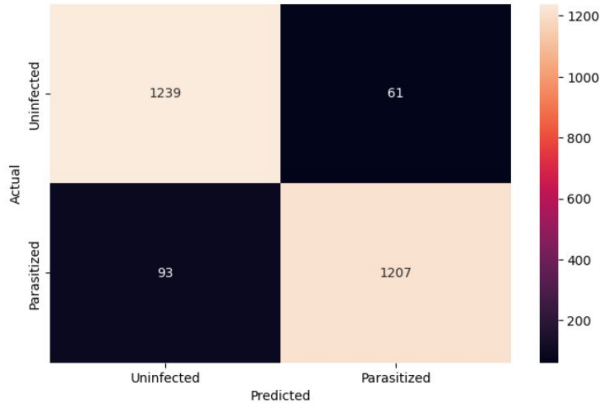
	precision	recall	f1-score
0	0.99	0.98	0.99
1	0.98	0.99	0.99

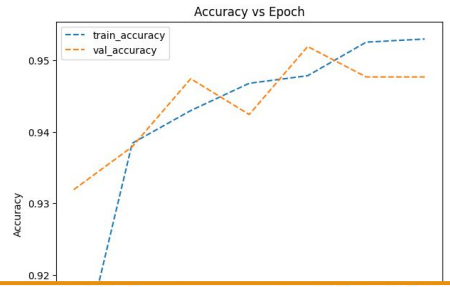
Appendix: Model 4

Observations: 97.5% test accuracy, lowest recall; over rely on Image -Net; over regularized model too heavy for data size; 14x params

```
transfer_layer = vgg.get_layer('block5_pool')

vgg.trainable = False

x = Flatten()(transfer_layer.output)

x = Dense(256, activation = 'relu')(x)

x = Dense(128, activation = 'relu')(x)

x = Dropout(0.3)(x)

x = Dense(128, activation = 'relu')(x)

x = BatchNormalization()(x)

pred = Dense(2, activation = 'softmax')(x)

model4 = Model(vgg.input, pred)
```

Confusion matrix





	precision	recall	f1-score
0	0.93	0.95	0.94
1	0.95	0.93	0.94