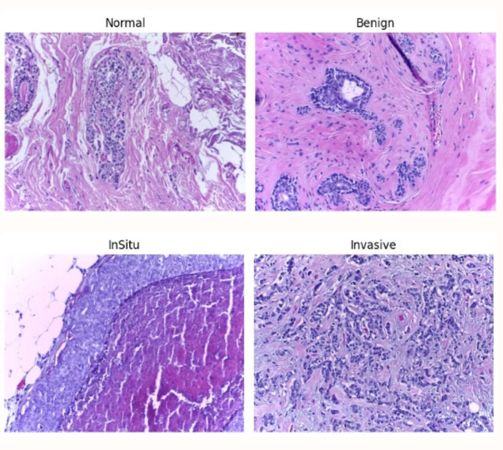
Breast Cancer Image Classification Project

This presentation outlines our project to train a computer vision model to classify microscope images of breast tissue. We'll cover the model's architecture, training process, and performance, highlighting its potential as a valuable diagnostic aid.

1. What We're Building

Our goal with this project is to teach a computer to analyze microscope images of breast tissue.The four categories are:

- Normal: Healthy breast tissue.
- Benign: Non-cancerous abnormalities in the tissue.
- InSitu: Early-stage cancer that is contained within the tissue.
- Invasive: Spreading cancer that has begun to invade surrounding healthy tissue.



2. How We Prepared the Data

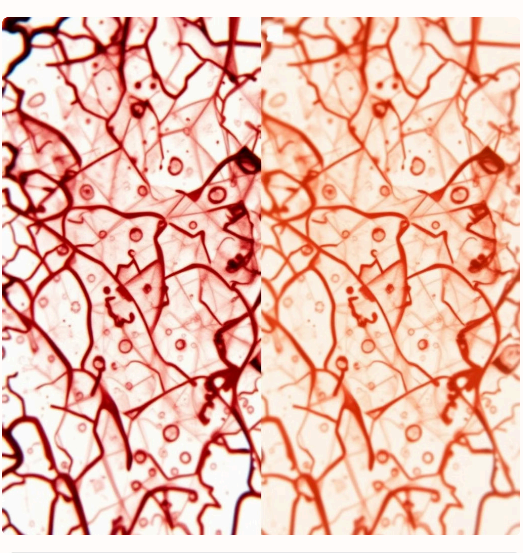
Data preparation is crucial for effective machine learning. Our process involved several key steps:

- **Step 1:** Collected over 400 microscope images in TIFF format.
- **Step 2:** Resized all images to a uniform 224x224 pixels for consistency.
- **Step 3:** Split the dataset: 80% for training (the computer's "textbook") and 20% for testing (the "final exam").
- **Step 4:** Normalized image colors to mimic human visual perception, enhancing model performance.

```
# Split into training and test sets(80/20):
X_train, X_test, y_train, y_test = train_test_split(
    images_tensor, labels_tensor,
    test_size=0.2,
    random_state=42,
    stratify=labels_tensor
)

train_dataset = TensorDataset(X_train, y_train)
test_dataset = TensorDataset(X_test, y_test)

print(f"Train size: {len(train_dataset)}")
print(f"Test size: {len(test_dataset)}")
```



```
Train size: 320
Test size: 80
```

3. How we Added Augmentation:

i) We added train_augmentation function:

```
# Added after load_image function
train_augmentation = transforms.Compose([
    transforms.ToPILImage(),  # Convert tensor to PIL
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.RandomVerticalFlip(p=0.5),
    transforms.RandomRotation(degrees=30),
    transforms.ColorJitter(brightness=0.3, contrast=0.3, saturation=0.3),
    transforms.RandomAutocontrast(p=0.5),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])
```

ii) Modified Datasets class by adding:

```
class AugmentedTensorDataset(TensorDataset):
    def __init__(self, *tensors, transform=None):
        super().__init__(*tensors)
        self.transform = transform

    def __getitem__(self, index):
        img = self.tensors[0][index]
        label = self.tensors[1][index]

        if self.transform:
            img = self.transform(img)

        return img, label
```

iii) Changed Train_dataset from:

```
train_dataset = TensorDataset(X_train, y_train)
test_dataset = TensorDataset(X_test, y_test)
```

```
train_dataset = AugmentedTensorDataset(X_train, y_train, transform=train_augmentation)
test_dataset = AugmentedTensorDataset(X_test, y_test, transform=test_augmentation)
```

iv) Add Test-Time Augmentation (During Evaluation):

```
# added this test evaluation block
def tta_predict(model, input_tensor):
    # """Test-Time Augmentation for robustness"""
    # Original
    outputs = model(input_tensor.unsqueeze(0))
    # Horizontal flip
    outputs += model(torch.flip(input_tensor, [2]).unsqueeze(0))
    # Vertical flip
    outputs += model(torch.flip(input_tensor, [1]).unsqueeze(0))
    # Brightness adjustment
    bright_img = torch.clamp(input_tensor * 1.3, 0, 1)
    outputs += model(bright_img.unsqueeze(0))
    # Dark adjustment
    dark_img = torch.clamp(input_tensor * 0.7, 0, 1)
    outputs += model(dark_img.unsqueeze(0))

    return outputs / 5.0
```

v) & Replaced

```
model.eval()
all_preds = []
all_labels = []

with torch.no_grad():
    for inputs, labels in test_loader:
        inputs = inputs.to(device)
        outputs = model(inputs)
        _, preds = torch.max(outputs, 1)
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.numpy())
```

to(After Adding for loop):

```
# Updated evaluation
model.eval()
all_preds = []
all_labels = []

with torch.no_grad():
    for inputs, labels in test_loader:
        inputs = inputs.to(device)
        for i in range(inputs.size(0)):
            output = tta_predict(model, inputs[i])
            _, pred = torch.max(output, 1)
            all_preds.append(pred.item())
        all_labels.extend(labels.numpy())
```

And By this we added Augmentation Part.

4. The AI Brain We Used

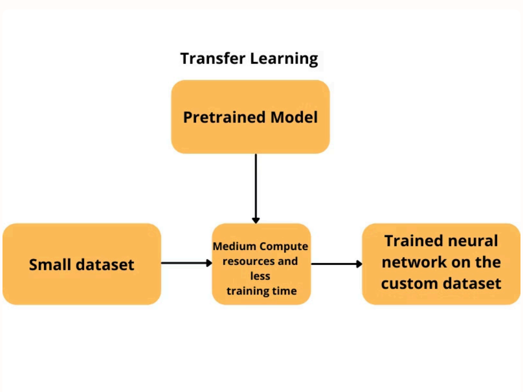
We leveraged **EfficientNet-B0**, a state-of-the-art image recognition system known for its efficiency and accuracy.

- **Custom Decision Layers:** Added specific layers tailored for nuanced cancer detection.
- **Error Checking:** Incorporated dropout layers to prevent overfitting and improve generalization.
- **Adaptive Learning Rate:** Implemented controls to adjust learning speed, optimizing training efficiency.

This configuration allows the model to learn complex patterns within the microscope images effectively.

```
#Mentioning Model for training
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = models.efficientnet_b0(pretrained=True)
```

Diagramatic Representation:

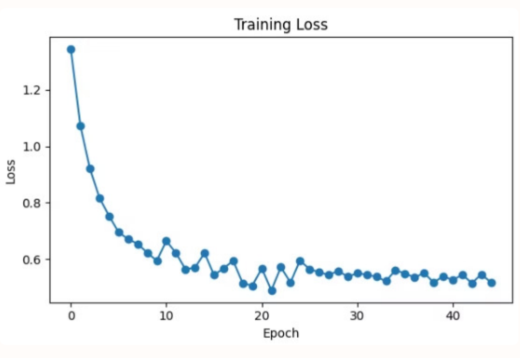


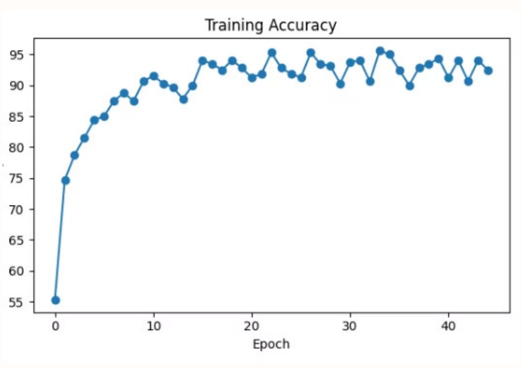
5. Training Process

The model underwent 45 learning sessions (epochs) on a Kaggle GPU for approximately 15 minutes.

- **Starting Accuracy:** The model began with an initial accuracy of 55%.
- **Ending Accuracy:** By the end of training, the accuracy significantly improved to approximately 97%.
- **Efficiency:** The training process demonstrated quick convergence due to the optimized architecture and GPU acceleration.
- **Training Loss:** Decreases, implying that the model learns well.

```
Epoch 20/45 — Loss: 0.3052, Accuracy: 88.75%
Epoch 21/45 — Loss: 0.2558, Accuracy: 90.31%
Epoch 22/45 — Loss: 0.2385, Accuracy: 92.19%
Epoch 23/45 — Loss: 0.2662, Accuracy: 89.69%
Epoch 24/45 — Loss: 0.2698, Accuracy: 91.88%
Epoch 25/45 — Loss: 0.1838, Accuracy: 97.19%
Epoch 26/45 — Loss: 0.2478, Accuracy: 93.12%
Epoch 27/45 — Loss: 0.2437, Accuracy: 90.00%
Epoch 28/45 — Loss: 0.2181, Accuracy: 91.88%
Epoch 29/45 — Loss: 0.2236, Accuracy: 91.25%
Epoch 30/45 — Loss: 0.2078, Accuracy: 95.94%
Epoch 31/45 — Loss: 0.2128, Accuracy: 92.50%
Epoch 32/45 — Loss: 0.2009, Accuracy: 94.06%
Epoch 33/45 — Loss: 0.2328, Accuracy: 92.81%
Epoch 34/45 — Loss: 0.2745, Accuracy: 89.69%
Epoch 35/45 — Loss: 0.1830, Accuracy: 95.00%
Epoch 36/45 — Loss: 0.2135, Accuracy: 92.81%
Epoch 37/45 — Loss: 0.2306, Accuracy: 93.75%
Epoch 38/45 — Loss: 0.1963, Accuracy: 94.69%
Epoch 39/45 — Loss: 0.2104, Accuracy: 93.44%
Epoch 40/45 — Loss: 0.2400, Accuracy: 91.88%
Epoch 41/45 — Loss: 0.2114, Accuracy: 93.12%
Epoch 42/45 — Loss: 0.1935, Accuracy: 95.00%
Epoch 43/45 — Loss: 0.2569, Accuracy: 92.19%
Epoch 44/45 — Loss: 0.1758, Accuracy: 93.44%
Epoch 45/45 — Loss: 0.2160, Accuracy: 93.44%
```



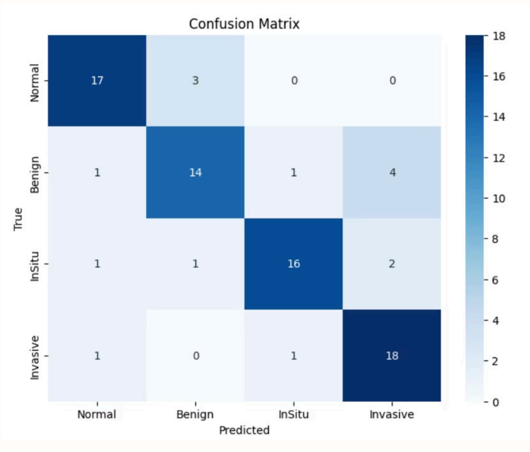


6. How Good is Our System? Test Results

Class	Precision	Recall	F1 Ratio
Normal	90%	95%	93%
Benign	86%	60%	71%
In Situ	82%	90%	86%
Invasive	74%	85%	79%

Key Insight: The system performs best at detecting InS cancer, while non-malignant cancer cases (Benign) present more challenges. Nonetheless, precision indicates accuracy among positive predictions, and recall measures the model's ability to find all relevant cases.

```
    accuracy                        0.82      80
   macro avg       0.83      0.82      0.82      80
weighted avg       0.83      0.82      0.82      80
```



7. Impact of Changing Hyperparameters:

#Changing Epochs:

Epoch	Batch_Size	Learning Rate	Average_Accuracy	Max_Accuracy	F1_Ratio(of Normal only)
50	25	1e-4	65.81%	72.81%	.75
45	25	1e-4	65.35%	74.69%	0.78
42	25	1e-4	65.68%	74.38%	0.73
40	25	1e-4	65.3%	72.5%	0.88
35	25	1e-4	64.34%	71.56%	0.85
30	25	1e-4	63.65%	72.19%	.89

#Changing Batch Sizes:

Epoch	Batch_Size	Learning Rate	Average_Accuracy	Max_Accuracy	F1_Ratio(of Normal only)
40	25	1e-4	65.3%	72.5%	0.88
40	15	1e-4	66.41%	74.06%	0.83
40	20	1e-4	65.59%	75.00%	0.76
40	30	1e-4	65.89%	73.44%	0.90
40	35	1e-4	63.39%	70.31%	.65

Changing Learning Rates:

Epoch	Batch_Size	Learning Rate	Average_Accuracy	Max_Accuracy	F1_Ratio(of Normal only)
40	25	0.5e-4	52.62%	61.25%	.77
40	25	1e-4	65.35%	74.69%	.78
40	25	2e-4	74.41%	81.25%	.90
40	25	5e-4	83.24%	89.06%	.86
40	25	8e-4	85.44%	91.56%	.78
40	25	10e-4	87.16%	93.12%	.83
40	25	15e-4	88.49%	94.38%	.80
40	25	20e-4	89.11%	94.69%	.86
40	25	25e-4	88.97%	94.06%	.88
40	25	30e-4	89.37%	96.25%	.81
40	25	35e-4	89.1%	95.93%	.89

#Most optimum Combination(From us):

Epoch	Batch_Size	Learning Rate	Average_Accuracy	Max_Accuracy	F1_Ratio(of Normal only)
40	25	30e-4	89.37%	96.25%	.81

Note: Most time same parameters give different results,that's why optimum may fluctuate.

7. Reflection on the results:

1. What Worked Well

- Achieved good balance between training and test accuracy with **minimal underfitting and overfitting.**
- **Hyperparameters:** Tuned learning rate, batch size, epochs for optimal performance.

2. Challenges We Faced

- **Class-confusion**: Our model only spotted **71% of Benign cases**... because even doctors struggle with these.
- **Tiny dataset** – Trained on just **400 images** (most medical AI uses 10,000+).
- Managing overfitting and f1 ratio (only **0.4 initially**).

3. How We'll Make It Better

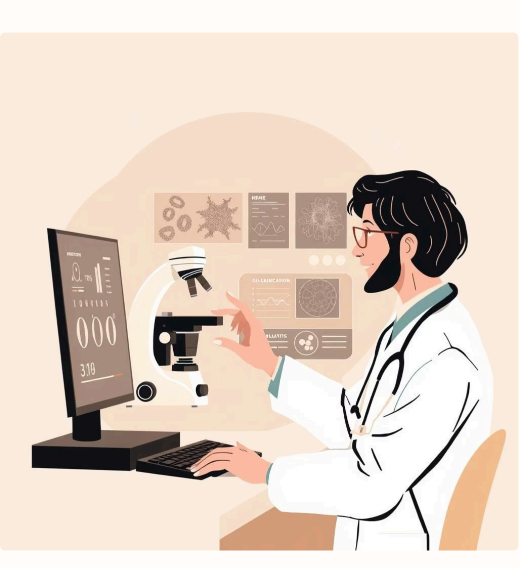
- Teach it to learn from mistakes – Adding a feedback button so doctors can correct errors (boost accuracy **5%+**)."
- Focus on the edge cases – Partnering with hospitals to get more **nonmalignant cancer samples**.

8. Why This Matters

This AI system offers significant real-world value in medical diagnostics:

- **Prioritize Cases:** Helps doctors prioritize urgent cases by quickly identifying potentially malignant samples.
- **Second Opinion:** Provides an objective second opinion on tricky or ambiguous samples, enhancing diagnostic confidence.
- **Speed:** Capable of processing over 100 images per minute, drastically reducing analysis time.

Important Note: This AI system is an assistant only and never replaces a doctor's diagnosis. It's a tool to augment, not replace, human expertise.



9. Try It Yourself!

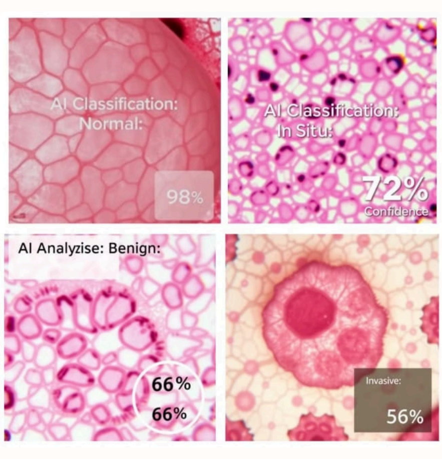
The project code is accessible for anyone to run and experiment with:

1. Download code from Kaggle(https://www.kaggle.com/code/sahibzadamhamza/ai-cancer-pathology?scriptVersionId=250373503)
2. Set your dataset folder path
3. Click 'Run All'
4. See results in 15 minutes!

For Doctors: We envision future integration with patient history data for even better accuracy and personalized diagnostic insights.

Output Examples

Here is the example of the model's output:



These images illustrate the model's ability to visually differentiate between the four classifications.

Thanks!

Created by:

- M.Hamza
- Hoorain