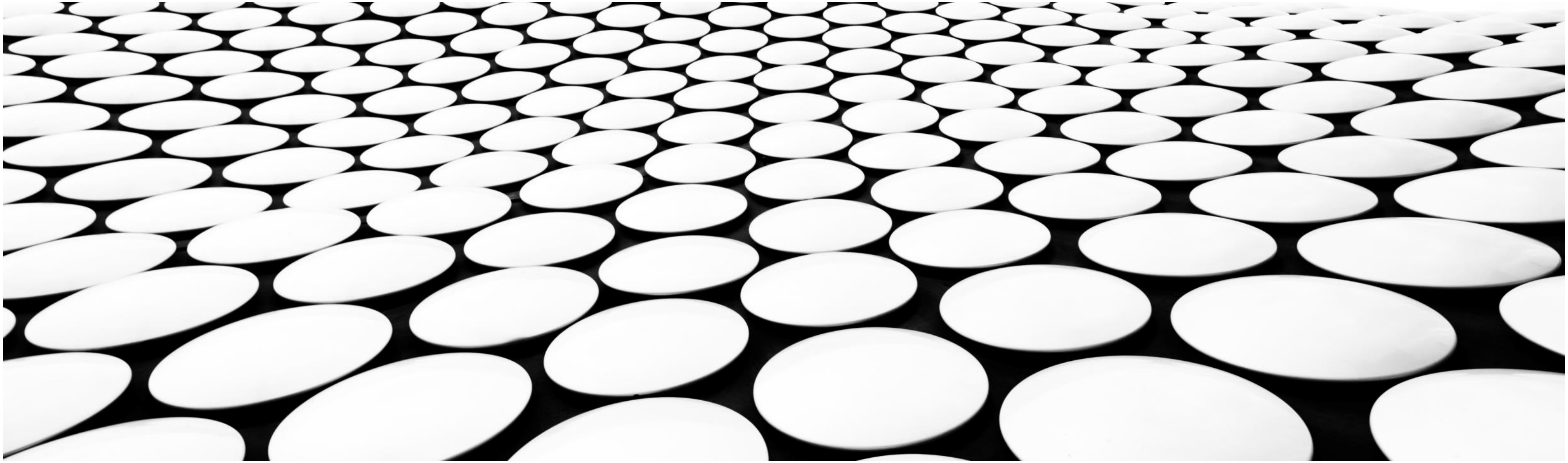
DSTI DEEP LEARNING PROJECT – COMPUTER VISION

SKIN CANCER DETECTION



PLAN ON A PAGE

Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Problem Analysis	Data Set Collection	Data Annotation and Labelling	Model Selection & Training	Model Evaluation and Iteration	Final Iteration & Deployment

Model testing, full team (Week 5)

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6
Data Science	**Models Review**	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**
	• Review of Hugging face and other sources • Top 5 recommendation	• Data set review vs top 5 models • Model reprioritisation	• Initial Feature Selection & Engineering • Model reprioritisation (if required)	• Model training & evaluation – base data set • Evaluation report • Iteration strategy, e.g. what to try next		• Ready Model for Kaggle Notebook (Production) deployment
Data Engineer	**Techniques Review**	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**
	• Review of image manipulation techniques • Top 5 recommendation	• Data set review • Storage selection • Data Processing • Data Cleansing	• <u>Additional Data Sets</u> • Data set review • Storage selection • Data Processing • Data Cleansing	• Ad hoc data engineering support for model training • Pick-up DS tasks as needed		• Ready data for Kaggle Notebook (Production) deployment
Data Analysis & Platform Engineering	**Platform Analysis**	**Week 2**	**Week 3**	**Week 4**	**Week 5**	**Week 6**
	• Review of Kaggle Notebooks • Review & recommend dev environment, CoLab, AWS, Local, etc	• Platform configuration • Identification of additional data sets	• <u>Data Analysis & Exploration</u> • Statistical Exploration (label count, etc.) • Data Viz (pair plots, histograms, etc)	• Model analysis and visualisation support • 1st Draft support Kaggle competition material (viz, slides, report, etc.)		• Ready support material for Kaggle competition submission

PLATFORM ANALYSIS

DEVELOPMENT ENVIRONMENTS

Feature	Google Colab	Local Environment	Kaggle Notebooks
Cost	Free (with paid options)	One-time hardware cost	Free
Ease of Use	Very easy	Moderate	Very easy
GPU Access	Yes	Depends on hardware	Yes (limited)
Scalability	Limited	Limited by hardware	Limited
Integration	Google Drive	Full control	Kaggle datasets & competitions
Runtime Limits	12 hours max	No limits	9 hours/week (free tier)
Pre-installed Libraries	Some	Manual installation	Extensive data science libraries
Data Access	Flexible	Local storage	Direct access to Kaggle datasets
Collaboration	Good	Limited	Excellent for competitions
Persistence	Session-based	Persistent	Version control included

DATA STORAGE OPTIONS

Feature	GitHub	AWS S3	Google Cloud Storage	Kaggle Datasets
Cost	Free for public	~$0.023/GB/month	~$0.020/GB/month	Free up to 20GB
Performance	Average	Excellent	Excellent	Good
File Size Limit	100MB per file	Virtually unlimited	Virtually unlimited	20GB total
Integration	Version control	AWS services	Google services	Kaggle platform
Ease of Use	Easy	Moderate	Moderate	Very easy
Scalability	Limited	Highly scalable	Highly scalable	Limited

KAGGLE VS. COLAB NOTEBOOKS

	Kaggle Notebooks	Google Colab Notebooks
Primary Use	Data science competitions	General-purpose
Environment	Pre-configured	Requires more setup
Data Access	Direct access to Kaggle datasets	Flexible, requires setup
Collaboration	Competition-focused	General collaboration
Runtime	Limited	Up to 12 hours
Integration	Kaggle ecosystem	Google ecosystem
GPU quota	30 hours per week (in the free tier)	Unlimited in theory, but with fair usage limits
GPU types	Usually Nvidia P100	Varies, can include T4, P100, V100
GPU assignment	Automatic GPUs assignment based on availability	Manual selection by the user
GPU Availability	Consistent performance with reliable GPU	Inconsistent: GPUs might not always be available, performance can vary between sessions

SETTING UP KAGGLE PROJECT

	Description	
Create or Connect Dataset	Upload and configure data	
Create Notebook	Set up coding environment	
Configure Notebook	Access data, import libraries	
Develop Model	Code, train, and test model	
Create Submission	Format results	
Submit Results	Upload to competition	

TRANSITIONING FROM KAGGLE TO COLAB/AWS

Aspect	Change Required	Complexity	Key Considerations
Code Transfer	Minor adjustments	Low	Data access parts need changes
Environment Setup	Manual setup in Colab	Medium	Install required libraries
Data Storage	Move to AWS S3	High	Security, costs, performance
Memory Management	Potential adjustments	Medium	Colab may have different limits
Runtime	Adjust for longer sessions	Low	Colab offers up to 12-hour sessions
Persistent Storage	Implement solution	Medium	Use Google Drive or re-download
Version Control	Set up GitHub integration	Medium	Replace Kaggle's built-in system
Result Submission	Manual process	Low	Download from Colab, upload to Kaggle

MODELS REVIEW

BINARY CLASSIFICATION ALGORITHM REVIEW

	Convolutional Neural Network (CNN)	Transfer Learning with Pre-trained Models (e.g., ResNet, EfficientNet)	Ensemble Methods (e.g., Random Forest, Gradient Boosting)	Support Vector Machines (SVM) with Kernel Trick	Attention-based Models (e.g., Vision Transformer - ViT)
Characteristics	• Specialized for image processing • Automatically learns relevant features from images • Deep architecture allows for complex pattern recognition	• Utilizes pre-trained models on large datasets • Fine-tunes the model for specific task • Leverages learned features from diverse image datasets	• Combines multiple models to improve performance • Can use different types of base models • Often used with decision trees as base learners	• Finds optimal hyperplane to separate classes • Can use different kernel functions for non-linear separation • Focuses on maximizing the margin between classes	• Uses attention mechanisms to process image patches • Can capture long-range dependencies in images • Inspired by transformer architectures in NLP
Advantages	• Excellent performance on image classification tasks • Can capture spatial hierarchies in images • Robust to variations in input	• Requires less task-specific training data • Often achieves high performance quickly • Benefits from features learned on large, diverse datasets	• Generally provides better performance than single models • Reduces overfitting • Can handle different types of features	• Effective in high-dimensional spaces • Memory efficient • Versatile through different kernel functions	• Can achieve state-of-the-art performance on many vision tasks • Scales well with larger datasets and model sizes • Can potentially capture more global context than CNNs
Disadvantages	• Requires large amounts of labeled data • Computationally intensive • Can be prone to overfitting on small datasets	• May require fine-tuning of hyperparameters • Can be computationally expensive for very large models • May not be optimal if target task is significantly different from pre-training task	• May be computationally expensive • Can be more complex to interpret • Requires careful tuning of ensemble parameters	• Can be slow to train on large datasets • Sensitive to feature scaling • May require careful parameter tuning	• May require large amounts of data for training from scratch • Can be computationally expensive • May struggle with small objects or fine-grained details
Code example					

MOST POWERFUL ALGORITHM COMPARISON FOR IMAGE CLASSIFICATION

	EfficientNet	DenseNet	ResNet	Inception-ResNet	Vision Transformer (ViT)
Definition	CNN architecture using compound scaling to balance network depth, width, and resolution	CNN with dense connectivity pattern between layers	Deep CNN using residual learning	Combines Inception architecture with residual connections	Applies transformer architecture to image patches
Characteristics	• Compound scaling • Mobile Inverted Bottleneck Convolution blocks • Squeeze-and-Excitation blocks	• Dense connectivity • Feature reuse • Compact architecture	• Residual learning • Skip connections • Very deep architecture	• Multi-scale feature extraction • Residual connections • Inception modules	• Patch-based image processing • Self-attention mechanism • No convolutions
Advantages	• Excellent efficiency-accuracy trade-off • Scalable architecture • Strong performance on various tasks	• Efficient parameter usage • Strong feature propagation • Mitigates vanishing gradient	• Enables training of very deep networks • Widely applicable • Well-understood architecture	• High accuracy • Multi-scale feature capture • Benefits of both Inception and ResNet	• Captures global dependencies • Scales well to large datasets • Potential for cross-modal applications
Disadvantages	• Complex architecture • May require large datasets for full benefit	• Memory intensive during training • Complex dense connections	• Very deep variants can be computationally expensive	• Computationally expensive • Complex architecture	• Requires large datasets for training from scratch • May struggle with small objects
Family Models	B0, B1, B2, B3, B4, B5, B6, B7, L2, V2 (S, M, L)	DenseNet-121, 169, 201, 264	ResNet-18, 34, 50, 101, 152, 50V2, 101V2, 152V2	Inception-ResNet-v1, Inception-ResNet-v2	ViT-Base, ViT-Large, ViT-Huge
Comparison of Variants	B0 (smallest) to B7 (largest) Each step increases accuracy and complexity V2 improves training speed and accuracy	Deeper variants (201, 264) offer higher accuracy but more parameters 121 often good balance of efficiency and performance	Deeper variants (101, 152) offer higher accuracy V2 variants improve training stability	v2 generally preferred over v1 for better performance	Larger variants (Large, Huge) offer higher accuracy but require more data and compute

5 SELECTED MODELS FOR SKIN CANCER DETECTION

	EfficientNet-B7	ResNet-152V2	DenseNet-201	Inception-ResNet-V2	Vision Transformer (ViT)
	Compound scaling method to balance network depth, width, and resolution. B7 variant is one of the largest and most powerful	Residual connections, allowing for very deep networks. The 152-layer version with V2 improvements is a powerful model for image classification	Connects each layer to every other layer in a feed-forward fashion, which helps mitigate the vanishing gradient problem and encourage feature reuse	Combines the Inception architecture, which uses multiple filter sizes in each layer, with residual connections from ResNet	Excellent performance on image classification tasks and represents a different paradigm in computer vision (not a traditional CNN)
Characteristics	• Optimized architecture for both accuracy and efficiency • Achieved state-of-the-art performance on ImageNet • Uses compound scaling to systematically scale network dimensions	• Very deep architecture with residual connections • Improved training stability for deep networks • Strong feature extraction capabilities	• Dense connectivity pattern between layers • Requires fewer parameters than traditional CNNs • Encourages feature reuse throughout the network	• Combines Inception modules with residual connections • Very deep and wide network • Efficient use of computational resources	• Treats image patches as tokens and applies self-attention • Inspired by transformer models in NLP • Can capture long-range dependencies in images
Advantages	• Excellent performance-to-parameter ratio • Can capture fine-grained details important in medical imaging • Scalable architecture allows for different model sizes	• Proven architecture in many computer vision tasks • Can capture complex hierarchical features • Good trade-off between depth and computational efficiency	• Efficient parameter usage • Strong gradient flow throughout the network • Can capture fine-grained features effectively	• High accuracy on image classification tasks • Can capture features at multiple scales simultaneously • Benefits from both Inception and ResNet architectures	• State-of-the-art performance on many vision tasks • Can potentially capture more global context than CNNs • Scales well with larger datasets and model sizes

CODE EXAMPLE

```python
# EfficientNet-B7 model
from tensorflow.keras.applications import EfficientNetB7

from tensorflow.keras.layers import GlobalAveragePooling2D, Dense

from tensorflow.keras.models import Model


base_model = EfficientNetB7(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

x = GlobalAveragePooling2D()(base_model.output)

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

output = Dense(1, activation='sigmoid')(x)


model = Model(inputs=base_model.input, outputs=output)


for layer in base_model.layers:

    layer.trainable = False


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

```python
# Vision Transformer (ViT) model
import timm

import torch

import torch.nn as nn


class ViTForBinaryClassification(nn.Module):

    def __init__(self, num_classes=1):

        super().__init__()

        self.vit = timm.create_model('vit_base_patch16_224', pretrained=True)

        self.vit.head = nn.Linear(self.vit.head.in_features, num_classes)


    def forward(self, x):

        return torch.sigmoid(self.vit(x))


model = ViTForBinaryClassification()

criterion = nn.BCELoss()

optimizer = torch.optim.Adam(model.parameters(), lr=1e-4)
```

SCIENTIFIC REPORT ABOUT MELANOMA DIAGNOSIS

SUMMARY OF THE OPTIMIZED DEEP-CNN ARCHITECTURE WITH CUSTOM MINI-BATCH LOGIC AND LOSS FUNCTION

Source: https://www.nature.com/articles/s41598-021-96707-8

- Topic: binary melanoma classification system using deep learning that outperforms 157 dermatologists
- Methodology:
  - Model architecture:
    - An optimized CNN architecture based on DenseNet169
    - Retrained of fully connected layers
  - Model innovation:
    - Custom Loss Function to handle imbalanced data
    - Custom mini-batch logic to maintain a fixed ratio between classes
    - Real-time data augmentation
  - Optimization: Adam optimizer & implementation of cyclical learning rate
  - Experimentation: Comparison of three models: ORI (original), BON (with custom batch), BLF (with custom batch and loss function) on ISIC 2019 test set (Test-10) and MClass-D dataset
- Main results:
  - The BLF model achieves an AUC of 94.4%
  - Sensitivity of 85.0% and specificity of 95.0% with a prediction threshold of 0.5
  - Balanced performance: 90.0% sensitivity and 93.8% specificity with an adjusted threshold
  - Outperforms all 157 tested dermatologists on MClass-D

SUMMARY OF THE OPTIMIZED DEEP-CNN ARCHITECTURE WITH CUSTOM MINI-BATCH LOGIC AND LOSS FUNCTION

Source: https://www.nature.com/articles/s41598-021-96707-8

- Performance analysis:
  - The custom loss function improves the balance between sensitivity and specificity
  - Custom mini-batch logic improves training stability
  - DenseNet169 architecture proves more effective than InceptionV3 and ResNet50 for this task
- Implications and perspectives:
  - Potential for application in computer-assisted medical diagnosis
  - Possibility to extend the approach to other medical image classification tasks
  - Need for further research on custom loss functions and fully connected layer architecture
- Limitations:
  - Need for validation on external datasets and in real clinical conditions
  - Necessity to interpret model results for medical use
- Contribution to the state of the art:
  - First approach outperforming all tested dermatologists on the MClass-D dataset
  - New method for handling imbalanced datasets in medical image classification