

**Identifying skin cancer through the analysis of skin lesions using machine learning.**

| **AUTHORS** | **STUDENT ID** |
| --- | --- |
| Aditya Kumar | 1006300 |
| Chin Wei Ming | 1006264 |
| Caitlin Daphne Tan Chiang | 1006537 |

Singapore University of Technology and Design

60.001: Applied Deep Learning

Ms. Pritee Agrawal

April 18, 2024

### Introduction

### Motivation

The primary motivation for this project is to enhance early skin cancer detection capabilities amongst the general population. Skin cancer is one of the most common cancer types globally, and early detection of this can significantly increase the chances of successful treatment for patients. Traditional methods for skin cancer diagnosis such as visual examinations, biopsy and histopathological analysis, are all effective but costly and time-consuming. By leveraging machine learning technologies to analyze skin lesions, we can provide a faster, more accessible preliminary diagnostic tool.

### Problem Statement and Objective

The primary challenge that we aim to address in this research is: “**How might we develop an automated and reliable system to identify the presence or absence of skin cancer in individuals using limited information?”**

This problem statement covers three important aspects:

1. **Automation:** Individuals should be able to automatically detect skin lesions without the need for medical expert intervention.
2. **Reliability:** The detection system must be accurate and dependable, minimizing the risk of false diagnoses.
3. **Limited Information:** Individuals should not be required to provide extensive information for a preliminary diagnosis.

The research objective is as follows: We aim to develop a **predictive model** that can **accurately determine** whether an individual is **healthy or has potential skin cancer** from an uploaded image of a suspected skin lesion. Furthermore, for cases identified as potential skin cancer, the model should also **classify the type of lesion** to provide insights to the individual.

### Scope

* The model will analyze an uploaded image of a suspected skin lesion and predict whether they are indicative of skin cancer.
* If potential cancer is detected, the model will further classify the lesion into one of several types based on its image appearance: Benign Keratosis (bkl), Melanocytic Nevi (nv), Dermatofibroma (df), Melanoma (mel), Basal Cell Carcinoma (bcc), Actinic Keratosis (akiec), and Vascular Lesions (vasc).
* To enhance user understanding and trust in the model’s outputs, the system will generate a heatmap overlay on the skin image. This heatmap will visually highlight the specific areas that the model identifies as indicative of a lesion, providing users with insight into what parts of the image influenced the model's prediction.

### Constraints and Limitations

* **Limited Dataset**: This research only included one dataset for training the model, which includes a predefined set of skin lesion types with associated images. The dataset size and variety are fixed, which may limit the generalizability of the model to other types of lesions or demographic groups not well-represented in the dataset.
* **Limited Detection Types**: The model is designed to recognize only a specific set of lesion types as mentioned in the research scope. It will not be able to identify other types of skin lesions.
* **Lack of Interpretability**: The model cannot provide explanations and supporting details for its predictions. This limitation is critical in medical applications where understanding the basis for a diagnosis is important for subsequent decision-making.

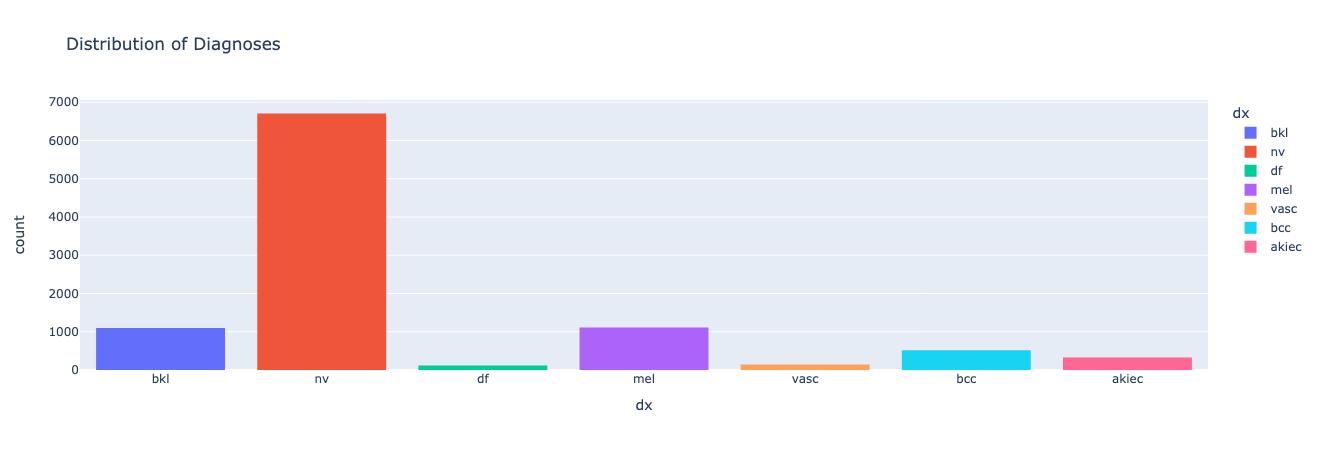
### Dataset

The dataset utilized is the HAM10000 dataset, sourced from Harvard Dataverse. Selecting the appropriate dataset has a large impact on how the model actually performs, as it fundamentally influences the model's predictive accuracy. The following details about the dataset are as follows:

1. **Images:** The dataset comprises 10,015 dermatoscopic images. All images are of different pixel sizes and are predominantly in JPEG format. This format is commonly used in medical image datasets due to its balance of quality and file size.
2. **Demographics**: The images are gathered from skin cancer clinics worldwide, including Austria, Australia, and the United States. The patients featured in the dataset do not belong to any specific age group, but are mostly Caucasian in terms of skin type. This lack of diversity will affect the overall data that the model is exposed to, and thereby negatively impact its accuracy.
3. **Metadata**: Each image is annotated with details such as the type of skin lesion, the bodily location of the lesion, and the patient's age and gender.

#### Data Augmentation

We decided to examine the distribution of classes in the original dataset. The results are as follows:



**Figure 1: Original Distribution of Skin Lesion Type Classes from the Original HAM1000 Dataset**

The chart in Figure 1 revealed a skewed distribution towards the lesion type of Melanocytic Nevi (nv). Such drastic imbalance such as this could potentially hinder the model’s training effectiveness due to bias in more frequent classes.

Hence, we implemented a data augmentation strategy to incorporate checks on the number of images per class. Our approach will emphasize the increase of representation of the underrepresented classes, aiming for a more balanced distribution amongst all skin lesion types. This adjustment can then enhance the model’s generalizations capabilities.

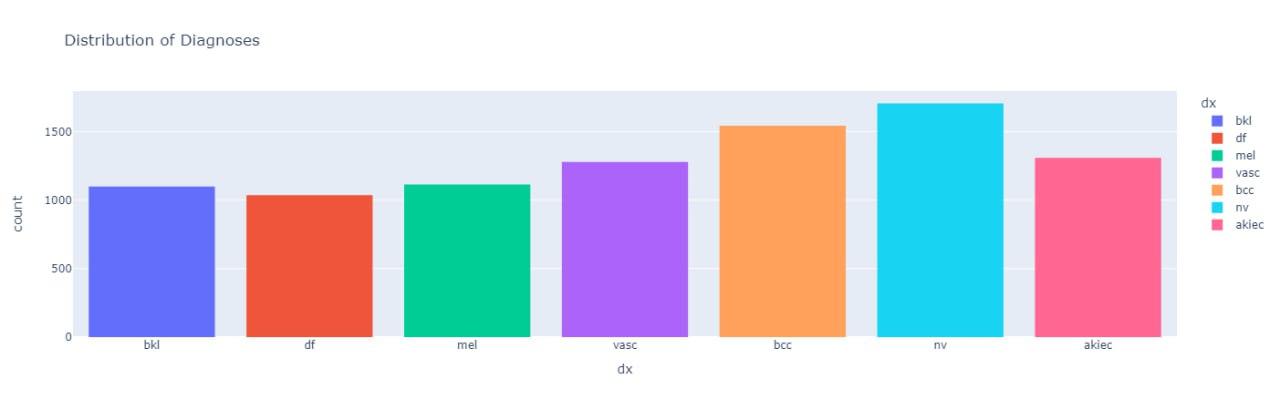
We utilized the ‘imgaug’ library in Python, a versatile package for image augmentation. The following augmenters from the package were then applied:

1. **Rotation**: Images are randomly rotated between -45 and 45 degrees.
2. **Blurring**: A gaussian blur is applied with a sigma value ranging from 0.0 to 3.0, to replicate the varying focus levels of the uploaded image.
3. **Noise Addition**: Gaussian noise is applied, with an intensity scale ranging from 10 to 60.
4. **Brightness Adjustment**: Brightness is either intensified or reduced by randomly scaling the pixel values, to expose the model to varying lighting conditions.
5. **Flipping**: Images are horizontally flipped to create a mirrored image.
6. **Cropping**: Random parts of the images are cropped out to represent different zoom levels and framing.
7. **Elastic Transformation**: Simulate the distortions that might occur during the imaging process.
8. **Perspective Transformation**: Mimic the effect of different camera angles.

We selected specific augmenters for each class based on their underrepresentation in the dataset, using the following configuration:

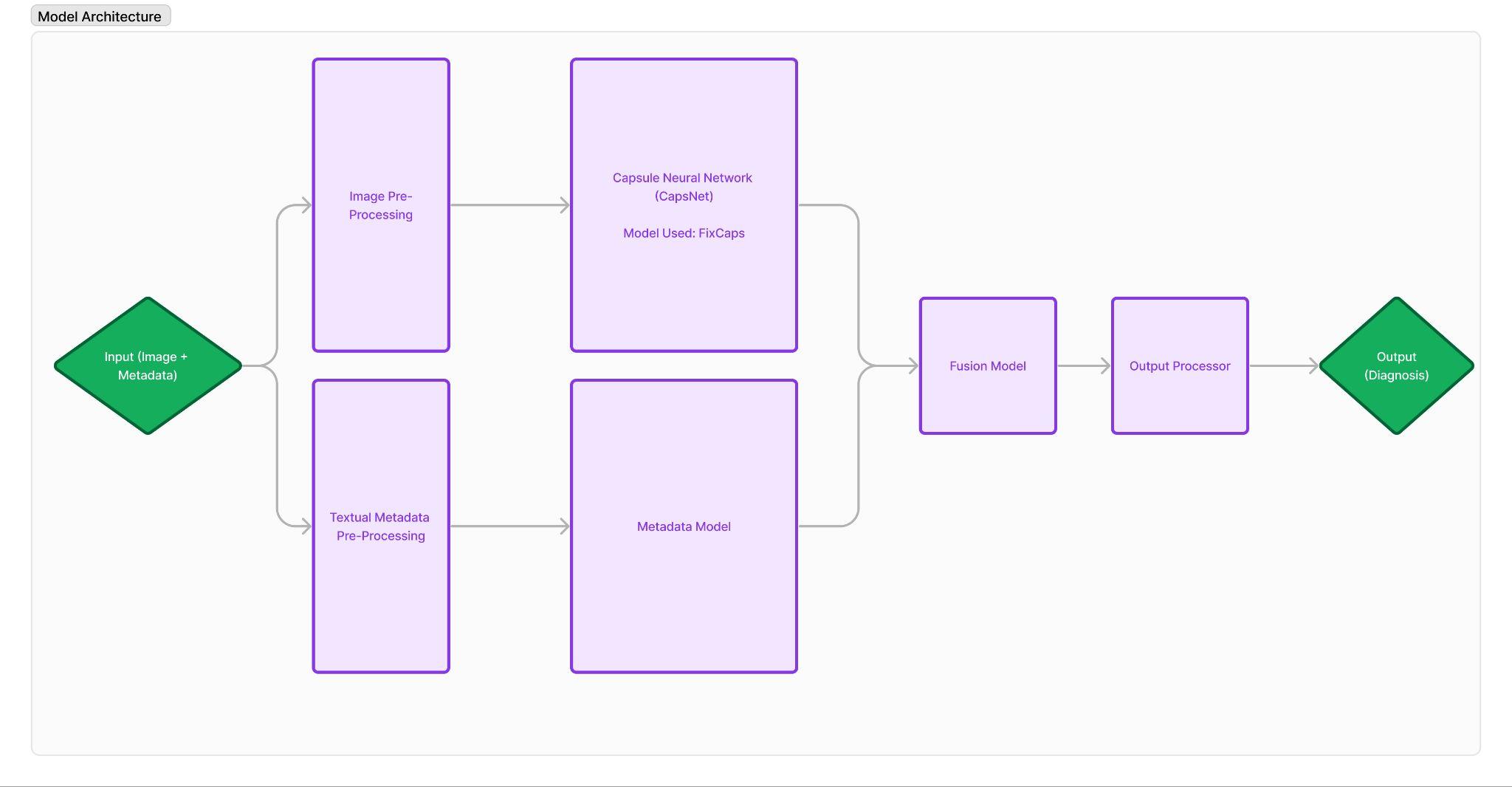
* **Basal cell carcinoma (BCC)**: Perspective and noise augmentations to enhance minor details often missed in smaller lesions.
* **Actinic keratosis (AKIEC)**: Blur, noise, and brightness adjustments to focus on enhancing texture and contrast variations.
* **Vascular lesions (VASC) and Dermatofibroma (DF)**: Application of all available augmenters to maximize variability due to their varied appearance.

Each image is then processed through its respective set of augmenters. The newly created images are saved and added to our augmented dataset, which ensures that all classes are equally represented. Figure 2 below showcases the improved and balanced distribution of classes in the adjusted dataset, which now consists of 9080 images.



**Figure 2: Revised Distribution of Skin Lesion Type Classes from the Augmented HAM1000 Dataset**

### Model Architecture



**Figure 3: Model Architecture**

As illustrated in Figure 2, the overall architecture of our predictive model comprises seven distinct components: the input layer, image pre-processing, textual metadata pre-processing, a metadata model, a Capsule Neural Network (CapsNet), a custom made fusion model, and an output processor.

#### Input Layer

The model accepts two types of input:

1. **Image Input**: A close-up image of the suspected skin lesion.
2. **Textual Input**: Demographic and anatomical data, specifically:

| **Attribute** | **Description** |
| --- | --- |
| age | The age of the individual |
| sex | The sex of the individual |
| localization | The specific body location of the suspected skin lesion |

**Table 1: Attributes and Descriptions of Model Inputs**

#### Image Pre-Processing

To ensure that the image is compatible with the capsule network’s architecture, several pre-processing steps are undertaken:

1. **Cropping:** The image is cropped to dimensions that comply with the input size requirements of the encoder.
2. **Normalization**: Pixel values are standardized across the image to a common scale, ensuring consistency with the data format used during model training.
3. **Tensor Transformation**: The normalized image is converted into a tensor format, the standard data structure used in capsule network frameworks.
4. **Device Allocation**: The tensor is transferred to a specific computational device in order to leverage hardware acceleration and speed up processing time.
5. **Batch Formation**: The tensor is incorporated into a batch containing a single item, as capsule networks are optimized to process data in batches.

After the pre-processing of the image has been completed, we then move on to the next step which is the pre-processing of the textual metadata input.

#### Textual Metadata Pre-Processing

To control the scope of possible metadata inputs to the model, the frontend of our application offers users three specific input fields:

1. A dropdown select for the patient’s sex
2. A number input for the patient’s age
3. A dropdown select containing a list of all the human body parts that skin lesions can appear on

Based on the patient’s input for each category, we convert these into numeric values via mapping. The specific values assigned between the attributes and its corresponding numeric value depends on the mappings established during training. This will be detailed in the ‘Dataset and Dataloader’ section of this paper.

#### Metadata Model

We have developed a custom metadata model class named ‘MetadataModel’. This model is integral to the Fusion Model portion of our architecture, in which its purpose is to integrate textual metadata into our classification pipeline. Here is an overview of how this model in particular operates:

1. First Layer: Transforms the three input features into a 128-dimensional space to capture complex data patterns. A ReLU (Rectified Linear Unit) activation is applied to introduce non-linearity to the model.
2. Dropout: Applies a 50% dropout rate to omit half the features during training iterations in order to avoid overfitting.
3. Second Layer: Further processes the data by reducing the dimensionality from the 128 space to 64. Another ReLU activation is applied to add further non-linearity to the model.

#### Capsule Neural Network (CapsNet)

The Capsule Neural Network (CapsNet) then receives the pre-processed image tensor as input.

In the early phases of this research, we initially focused on using a convolutional neural network (CNN). However, upon testing various models and conducting further research, we decided to adopt a more exploratory approach by utilizing the less common CapsNet architecture instead.

Unlike CNNs, CapsNet processes information using a hierarchical grouping of neurons known as "capsules." These capsules are designed to encode and preserve spatial hierarchies between features, allowing the network to recognize objects and their variations with better generalization. Hence, it would theoretically work better than regular CNNs in tasks such as image classification. (Mensah, Adekoya, Ayidzoe, & Edward, 2022).

The CapsNet then has two outputs:

1. A tensor that assigns classification scores to each of the 7 types of skin lesions from the dataset, reflecting the probability of each possible diagnosis.
2. A feature map output used for visual explanations via Grad-CAM.

The specifics of the network, including the number and size of layers, will be determined during evaluation.

#### Fusion Model

We have conducted a hypothesis that we can further enhance the accuracy of our CapsNet model if we were to add in the metadata context of the image. Hence, we have developed a custom fusion model that integrates data from two distinct sources: the CapsNet model and the MetadataModel class.

The following is the general structure of the fusion model:

1. **Feature Extraction**: The model extracts relevant features from the primary output of the CapsNet model.
2. **Metadata Feature Extraction**: It also extracts relevant features from the MetadataModel.
3. **Feature Combination**: The extracted features from both sources are then combined into a single feature set.
4. **Classification**: This combined feature set is processed through an AdvancedClassifier model.

The AdvancedClassifier is a neural network designed to integrate and classify input data effectively. The following describes how the overall structure operates:

1. The classifier expands the input into a higher dimensional space, followed by a ReLU activation to introduce non-linearity. Furthermore, a dropout is used in order to prevent overfitting.
2. A residual connection is used to add the original input directly to the output of the later layers, helping to preserve possible lost information described by the vanishing gradient problem.
3. The final layer maps the processed features into class logits, representing the prediction scores of the various classes.

The fusion model then has two outputs:

1. The class logits returned by the final layer of the AdvancedClassifier.
2. The feature map from the CapsNet model’s secondary output.

#### Output Processor

The output processor module handles the conversion of the two outputs from the Fusion model into a user-friendly format.

1. **Classification of Skin Lesions**

The first function of the output processor module is to interpret the tensor of class logits provided by the Fusion model. The function identifies the index of the highest score within this tensor, which corresponds to the most likely skin lesion type predicted by the model. This numerical index is then mapped to its corresponding skin lesion class name.

Furthermore, a threshold of 0.4 is set. This means that if none of the class probabilities in the softmax output meets this threshold, the function will return a statement that no skin lesion is detected within the image.

1. **Generating a Heatmap for Grad-CAM**

The second function of the output processor module involves converting the feature map output from the Fusion model into a usable heatmap, using the Gradient-weighted Class Activation Mapping (Grad-CAM) technique. Grad-CAM enables the inspection of how a neural network predicts its output by using gradients. We are then able to gain insight into how our model used the input image given to make its prediction. The actual implementation of Grad-CAM will be discussed in the ‘Implementation’ section of this research.

### Experiments and Results

To determine the most effective model for this research, we have conducted a comparative analysis of the four most popular used ones in regards to skin lesion classification.

#### Chosen Models for Comparison

In the initial version of our experiment when we were still focused on the CNN architecture, we tested the following models: ResNeXt, DenseNet, and InceptionV3. This was conducted under the conditions of 50 epochs and a batch size of 32. The results are detailed in Appendix 1. Among these models, InceptionV3 achieved the poorest test accuracy of 71.36%.

However, we have decided to set this model as our baseline for comparison and benchmarking. Even if it did not perform the best accuracy, its optimized architecture and widespread use in image classification tasks provides a robust framework for assessing new models (Mujahid et al., 2022).

Therefore, we have conducted research on other possible encoders that deliver the highest accuracy in image classification tasks, particularly in the classification of skin lesion types.

**Vision Transformer (ViT)**

On top of exploring CNNs, we started to explore the Vision Transformer (ViT) architecture due to its advanced self-attention mechanisms. Unlike traditional CNNs, the ViT processes images as a sequence of patches, utilizing transformer mechanisms to capture complex patterns and relationships between these patches. This capability makes it excellent at handling the classification of skin lesions, achieving an accuracy of 94.1% (Yang, Luo, & Greer, 2023).

**Fixed Capsule Network (FixCaps)**

The Fixed Capsule Network presents a novel approach by encoding spatial hierarchies between features. This attribute is especially valuable in medical imaging, where the precise spatial arrangement and orientation of features are critical for accurate diagnoses. By structuring data in this hierarchical fashion, FixCaps can enhance the model’s ability to interpret complex medical images. It has the potential to reach an accuracy of 96.49% with the HAM10000 dataset (Chongqing Jiaotong University, 2022).

**EfficientNet**

EfficientNet offers a scalable and efficient architecture through its innovative compound scaling method. This method uniformly scales all dimensions of depth, width, and resolution, based on a set compound coefficient. This balanced scaling enables EfficientNet to maintain or increase accuracy whilst using fewer parameters than traditional CNNs, allowing for a more cost-effective application. The model that we have tested with, EfficientNet B0, has a potential accuracy of 83.02% (Ali, Shaikh, Khan, & Laghari, 2022).

#### Dataset and Dataloader

The models of InceptionV3, ViT, EfficientNet, and the original version of FixCaps involve the same data preparation and dataloader process. Here is how the flow is structured:

1. **Data Loading and Preparation**

The dataset metadata is loaded from a CSV file. The image paths are standardized, and the textual skin lesion type labels are converted into numerical form using a predefined mapping. Additionally, categorical metadata fields such as gender and localization are also encoded into numerical form. This is done by converting these fields into category data types and then using these categories' integer codes as inputs for the model. The age metadata field can be used directly as it is already in numerical format.

1. **Data Splitting**

The dataset is divided into three subsets with the following recommended splits: training (80%), validation (10%), and testing (10%) (Draelos, 2019). These splits are executed using the train\_test\_split function from the sklearn library in Python. The training set is utilized to train the model, the validation set to tune the model’s parameters, and the testing set to assess its final performance.

1. **Custom Dataset Class**

The HAM dataset class manages the loading and preprocessing of image data and their corresponding labels. It fetches the individual images, converts them to RGB, and applies the transformations of resizing and tensor conversion. Then, it prepares the labels as tensors, ensuring that each image-label pair is correctly formatted for neural network input.

1. **Dataloaders**

Instances of the HAM custom dataset class are created for each data subset: training, validation, and testing. To facilitate the training of neural networks, data loaders are set up for each dataset instance. These loaders handle the batching of images into sets of 32 (batch size), which significantly improves the efficiency of the training process. For the training dataset specifically, the loader also shuffles the data to help in improved generalization.

Upon utilizing the same data preparation and dataloader strategy with FixCaps, its accuracy results came out to be lower (to be discussed in the ‘Further Training of the FixCaps’ section of the research). We instead opted to use the dataloader structure proposed by the research from Chongqing Jiaotong University (2022). The changes in the revised dataloader structure that we have implemented for the improved version of our FixCaps model is described as follows:

1. **Image Transformations**

Training images are randomly resized and cropped to 299x299 pixels, subjected to random vertical flips for augmentation, and normalized with a mean and standard deviation of [0.5, 0.5, 0.5] across all color channels. In contrast, validation images are simply resized to 302 pixels before being centrally cropped to 299x299 pixels and normalized using the same parameters as the training images.

1. **Data Splitting**

The new ratio for the data split is 80% for training and 20% for validation.

1. **Dataloaders**

For training, the DataLoader is set with a batch size of 168, shuffling enabled, and four worker threads to enhance parallel processing and efficiency. For validation, the batch size is reduced to 40, with shuffling disabled and the same number of worker threads, ensuring stable and consistent data feed during model evaluations.

#### Training the Models

As parameters for the training functions used in each of the four models, the following parameters are chosen:

1. Optimizer: Adam

We chose to use Adam optimiser as it is adapted as a benchmark for deep learning algorithms and recommended as the default optimisation algorithm. Moreover, Adam optimiser is more straightforward to implement and has a faster run time. We did not have the computational resource to experiment with other optimizers and therefore decided to go forward with Adam optimiser.

1. Learning Rate: 0.005
2. Loss: Cross Entropy Loss

Cross Entropy Loss is widely used for training a multi-class classification deep neural network.

1. Batch Size: 32
2. Epochs: 50

**Training Function: InceptionV3, ViT, and EfficientNet**

The training function’s main task is to conduct 50 epochs of training and validation cycles. During each epoch, the function trains the model on batches of data, adjusts parameters using backpropagation, and tracks training accuracy and loss. It also evaluates the model against a validation set to monitor generalization capabilities, logging these metrics for analysis.

**Training Function: FixCaps**

Upon experimentation of utilizing the first training function, we have found that it did not yield accurate results. The values obtained were extremely low and were obvious errors.

The FixCaps model requires a specialized training approach due to its unique design features. The FixCaps network uses large-kernel convolutions and a convolutional block attention module (CBAM) to improve the receptive field and reduce spatial information loss. This is critical for capturing detailed spatial hierarchies in medical images​​ (Chongqing Jiaotong University, 2022).

The training function for FixCaps is adapted to include one-hot encoding of target labels and a custom loss calculation that aligns with its capsule dynamics. This function also manages vector magnitudes from the output layer to facilitate precise predictions. These training adjustments take advantage of the FixCaps structure and its ability to process detailed feature relationships (Chongqing Jiaotong University, 2022). Thus, proving to be a more effective method of training, compared with the more general methods used for CNNs and Vision Transformers​​.

#### Training Results

To assess the effectiveness of the training outcomes for each model, we performed a comprehensive analysis using various metrics to determine the best-fit model for our needs. Recognizing that training accuracy alone does not provide a complete picture of a model's performance, we identified additional metrics essential for a more holistic evaluation.

Additional Metrics:

1. **Accuracy**: Percentage of correctly predicted image classes to the total number of images
2. **Precision**: Number of true positives divided by the number of true positives plus the number of false positives
3. **F1 Score**: A harmonic mean of the precision and recall of a classification model. A high F1 score value would indicate high precision and high recall values.
4. **Recall**: Number of true positives divided by the number of true positives plus the number of false negatives

The detailed performance metrics for each model are provided in Appendix 2.

InceptionV3, which served as our baseline, displayed the weakest performance amongst all tested models. Consequently, we have narrowed down our selection to the three other models that we aimed to explore. To facilitate a direct comparison between the best performing model of the three (ViT) and the worst performing (FixCaps), we have prepared the following information:

| **Models** | **Training Accuracy** | **Training Loss** | **Val Accuracy** | **Val Loss** | **Test Accuracy** |
| --- | --- | --- | --- | --- | --- |
| ViT | 88.40% | 0.3255 | 79.92% | 0.6054 | 79.34% |
| FixCaps | 77.89% | 0.1487 | 77.24% | 0.1657 | 77.03% |

**Table 2: Training, Validation, and Test Accuracies for ViT and FixCaps**

| **Models** | **Accuracy** | **Precision** | **F1 Score** | **Recall** |
| --- | --- | --- | --- | --- |
| ViT | 79.34% | 78.36% | 78.58% | 79.34% |
| FixCaps | 77.03% | 65.00% | 49.00% | 46.00% |

**Table 3: Evaluation Metrics for ViT and FixCaps**

Based on the data presented, ViT emerges as the best-suited model for our intended task. However, it's important to note that FixCaps is recognized as the benchmark model and was anticipated to deliver the highest accuracy scores. Anomalies in our findings suggest that the comparison may not have been fair due to FixCaps not being trained under the appropriate conditions and parameters (Chongqing Jiaotong University, 2022). This discrepancy leads us to hypothesize that, if trained correctly, FixCaps could potentially outperform ViT. Currently, the performance metrics of FixCaps are close to those of ViT, even under suboptimal conditions.

In addition, the FixCaps model showcased the least validation loss, which could indicate that it performs well on data that it hasn’t been exposed to yet.

Hence, we decided to train FixCaps under the appropriate conditions and it ultimately proved our hypothesis to be valid as it achieved an accuracy of 86.43%.

#### Further Training of the FixCaps Model

Based on research from Chongqing Jiaotong University (2022), the most effective training parameters for FixCaps have been identified as follows:

1. Batch Size: 168

We experimented with batch sizes of 32, 64, and 168. However, the batch size of 168 showcased the best performance (Appendix 3).

1. Optimizer: Adam
2. Loss: Cross-Entropy Loss
3. Learning Rate: 0.123 (with CosineAnnealingLR scheduler)

We started with a considerably high learning rate of 0.123, making bigger steps in the gradient descent at the beginning stage of the training. By implementing a CosineAnnealingLR scheduler, we are able to decrease the learning rate to a minimum value from a large learning rate, before rapidly increasing it again. This strategy gained popularity in gradient based optimization, especially with image classification to increase the rate of convergence.

1. Epochs: 100

We first attempted to train the FixCaps Model with the 10,000 images from the original HAM10000 dataset, but now under its specific training parameter conditions. In addition, we have also incorporated the revised dataloader process as mentioned in the ‘Dataset and Dataloader’ section of the paper. The training took approximately 1.5 hours to complete with Google Colab’s V100 GPU.

Using the given parameters above and 10,000 images, the following results are achieved:

| **Training Accuracy** | **Training Loss** | **Val Accuracy** | **Val Loss** |
| --- | --- | --- | --- |
| 74.45% | 0.1662 | 81.82% | 0.1772 |

**Table 4: Training and Validation Accuracies for FixCaps with 10,000 Original Images**

As mentioned earlier in the ‘Data Augmentation’ section of the paper, we have created an augmented version of the dataset that consists of an even distribution. However, this did not end up yielding better results than the previous original 10,000 images, as seen in Table 5 below:

| **Training Accuracy** | **Training Loss** | **Val Accuracy** | **Val Loss** |
| --- | --- | --- | --- |
| 66.20% | 0.2330 | 69.77% | 0.24329 |

**Table 5: Training and Validation Accuracies for FixCaps with 9080 Augmented Images**

Therefore, we decided to introduce an additional dataset comprising 40,000 images with an even class distribution (Appendix 4). This new dataset is from (“HAM10000(Aug)\_数据集-飞桨AI Studio星河社区,” n.d.), wherein the creators of this dataset expanded upon the original HAM10000 by concatenating more images, both augmented and from other sources. We hypothesized that the limitation of an uneven class distribution in the original HAM10000 dataset was negatively contributing to the accuracy of the model, but perhaps our proposed augmentation strategy may have its limitations in its structure that also causes inaccurate predictions.

It is important to note that we are not able to incorporate metadata in this 40,000 version of the dataset as they were not provided. Nevertheless, this did not end up being a hindrance as metadata does not necessarily improve the model’s performance (to be discussed in the ‘Training the Fusion Model’ section of the paper).

Using the given parameters above and the enlarged 40,000 images dataset, the following results are achieved:

| **Training Accuracy** | **Training Loss** | **Val Accuracy** | **Val Loss** |
| --- | --- | --- | --- |
| 70.20% | 0.2117 | 86.43% | 0.1343 |

**Table 6: Training and Validation Accuracies for FixCaps with 40,000 Images**

| **Test Accuracy** | **Precision** | **F1 Score** | **Recall** |
| --- | --- | --- | --- |
| 80.81% | 80.61% | 80.61% | 80.69% |

**Table 7: Evaluation Metrics for Fixcaps with 40,000 Images**

#### Training the Fusion Model

The last version of the FixCaps model has demonstrated high accuracy and strong performance across various evaluation metrics. Nonetheless, in line with our model architecture, we hypothesized that a fusion model combining both metadata and image data could yield even more accurate results. Therefore, we decided to develop and train our fusion model as described in the 'model architecture' section of this paper.

Using the same training parameters specified for the more accurate version of Fixcaps, we applied the same training process to this fusion model. Here are the results:

| **Training Accuracy** | **Training Loss** | **Val Accuracy** | **Val Loss** |
| --- | --- | --- | --- |
| 72.43% | 0.7633 | 68.04% | 0.8370 |

**Table 8: Training and Validation Accuracies for the Fusion Model**

| **Test Accuracy** | **Precision** | **F1 Score** | **Recall** |
| --- | --- | --- | --- |
| 72.46% | 49.20% | 69.05% | 72.00% |

**Table 9: Evaluation Metrics for the Fusion Model**

We conclude that the fusion model, which integrates both metadata and image data, did not improve the accuracy demonstrated by the latest version of our trained FixCaps model. Instead, it showcased weaker results across all evaluation metrics. This underperformance may be caused by limitations and potential inaccuracies in our fusion model's architecture. Despite these findings, for exploratory purposes we have chosen to integrate this fusion model to the frontend portion of our application.

### Implementation

For usability of our application, we have decided to implement a frontend that connects to all our models in the backend. We aim to provide users with a clear and simple method to upload images, and receive the appropriate predictions. Here is the link to our web application: <https://skin-lesion-app.onrender.com/>

The actual implementation of all the various parts of our model is described in the model architecture section.

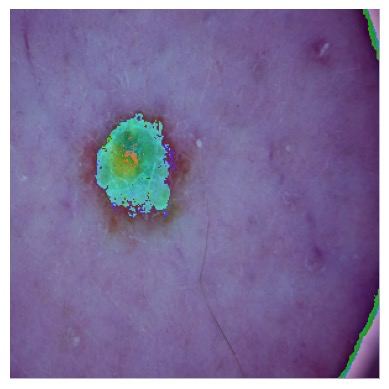
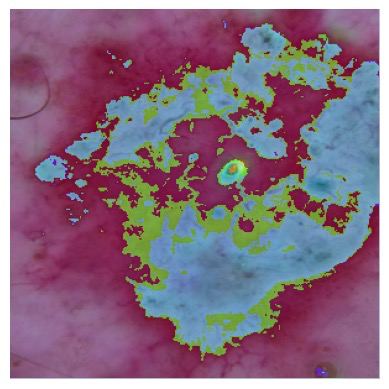
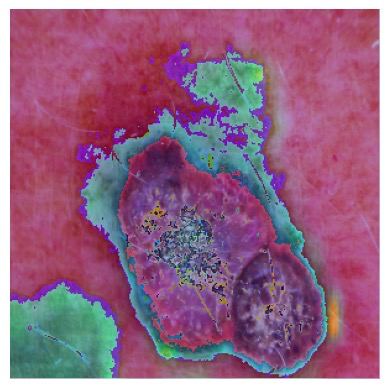
With the base structure developed, we are now able to apply specific dimensions in the image pre-processing section. The cropping of images will be to size 299x299, and the normalization will be set to transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)), meaning that each pixel value will be adjusted to have a mean of 0.5 and a standard deviation of 0.5.

As we have decided to proceed with the latest version of our trained FixCaps model, we then applied its architecture to be our image processing model. Here are the specifics of the model configuration we utilized, as they are the conditions at which the model performs the best (Chongqing Jiaotong University, 2022):

| **Parameter** | **Value** | **Description** |
| --- | --- | --- |
| n\_channels | 3 | Specifies the number of channels in the input image |
| n\_classes | 7 | Number of output classes the model can predict |
| conv\_outputs | 128 | Number of output channels produced by the initial convolutional layer(s) of the network |
| num\_primary\_units | 8 | Number of primary capsules in the Primary\_Caps layer of the capsule network.  This layer services as the initial capsule layer, converting convolutional outputs into capsule vectors |
| primary\_unit\_size | 576 | Size of each primary unit in the Primary\_Caps layer, which is determined by the output size of the previous convolutional layer(s) |
| output\_unit\_size | 16 | Size of the output vector from each capsule in the final capsule layer (Digit\_Caps) |
| img\_size | 299 | Size of the input images that the network expects |
| init\_weights | True | Determines whether to initialize the network weights using a specific method upon instantiation |

Then, with the integration of the FixCaps and MetadataModel modules into our Fusion model, we were able to proceed with testing and successfully receive the desired outputs. These outputs were then processed via our output processor module. Here are the final displays presented to the user in the frontend:

1. Classification of the skin lesion type, if detected, or a statement indicating that none were detected.
2. A heatmap image showing which parts of the skin the model focuses on most to generate its prediction. As can be observed from Figure 4 below, Grad-Cam successfully identifies the significant areas for each type of skin lesion.



**Figure 4: Examples of the Grad-Cam Output on 3 Different Class Sample Input Images**

### Conclusion

Using our custom-built fusion model, which integrates the output from the FixCaps model with our metadata model, we have successfully met the objectives of this research. The goal was to develop a predictive model capable of determining from an uploaded image whether an individual is healthy or potentially has skin cancer. Additionally, in cases of suspected skin cancer, it classifies the type of lesion to offer insights to the individual. The training accuracy for the fusion model reached 63.95%, while the FixCaps model achieved a training accuracy of 86.43%. There is definitely room for improvement in future works, but this research represents a step towards the use of AI for healthcare related tasks that is accessible to the common individual.

### Appendix

**Appendix 1: Initial CNNs Accuracy Comparison**

| **CNN** | **Test Loss** | **Test Accuracy** |
| --- | --- | --- |
| InceptionV3 | 1.0066 | 71.36% |
| DenseNet | 0.7242 | 74.45% |
| ResNeXt | 0.7029 | 79.74% |

**Appendix 2: Training and Evaluation Results of the Four Models**

| **Models** | **Training Accuracy** | **Training Loss** | **Val Accuracy** | **Val Loss** |
| --- | --- | --- | --- | --- |
| ViT | 88.40% | 0.3255 | 79.92% | 0.6054 |
| FixCaps | 77.89% | 0.1487 | 77.24% | 0.1657 |
| InceptionV3 | 68.80% | 1.1430 | 0.6953 | 1.0592 |
| EfficientNet | 79.86% | 0.5716 | 78.22% | 0.6506 |

| **Models** | **Test Accuracy** | **Precision** | **F1 Score** | **Recall** |
| --- | --- | --- | --- | --- |
| ViT | 79.34% | 78.36% | 78.58% | 79.34% |
| FixCaps | 77.03% | 65.00% | 49.00% | 46.00% |
| InceptionV3 | 71.36% | 73.66% | 69.69% | 71.36% |
| EfficientNet | 78.74% | 78.86% | 78.49% | 78.74% |

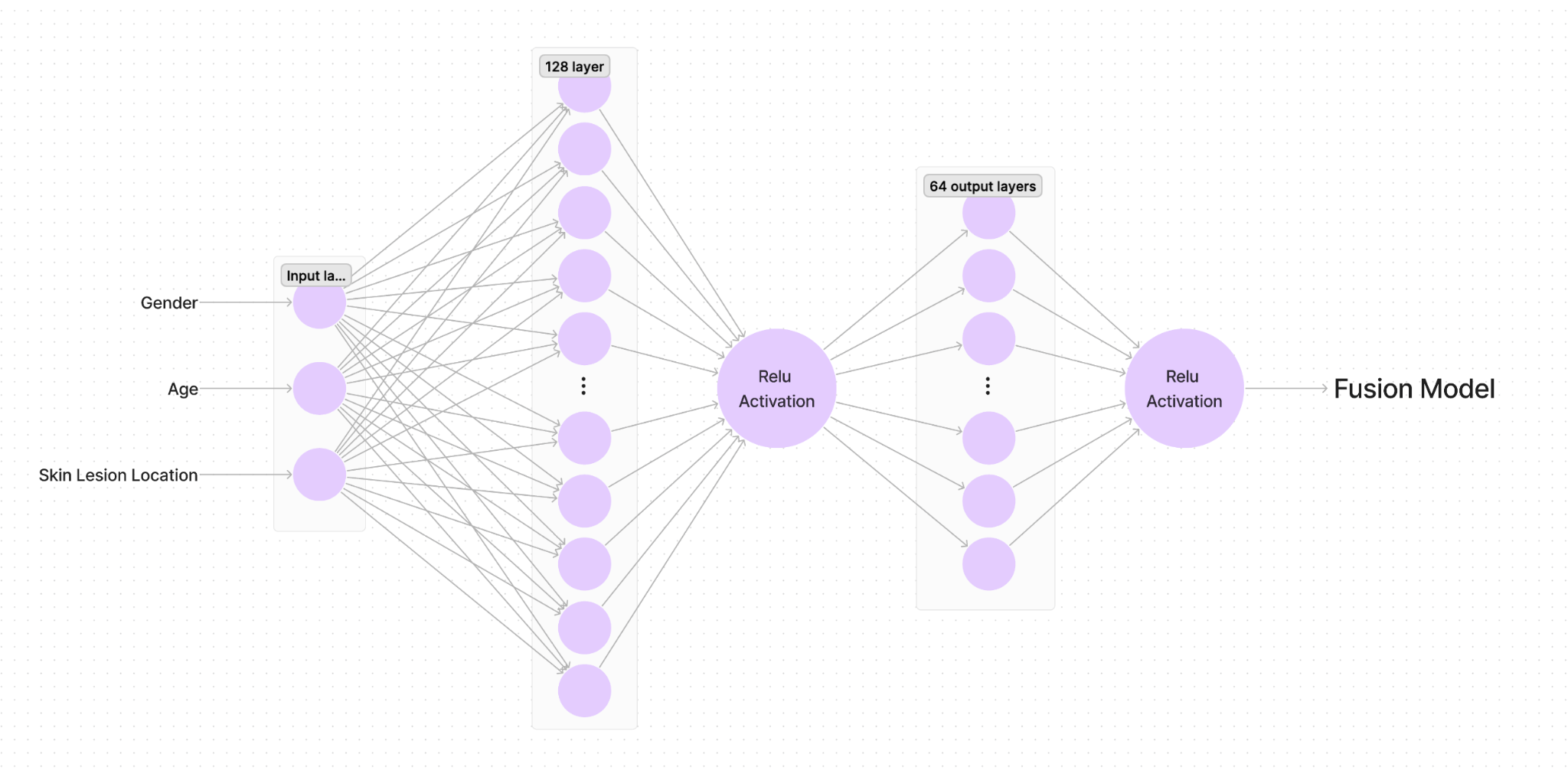
**Appendix 3: Batch Size 32, 64, and 168 and their performance**

| **Batch Size** | **Test Loss** | **Test Accuracy** |
| --- | --- | --- |
| 32 | 0.16536 | 77.23996 |
| 64 | 0.17702 | 77.47368 |
| 168 | 0.18469 | 80.60606 |

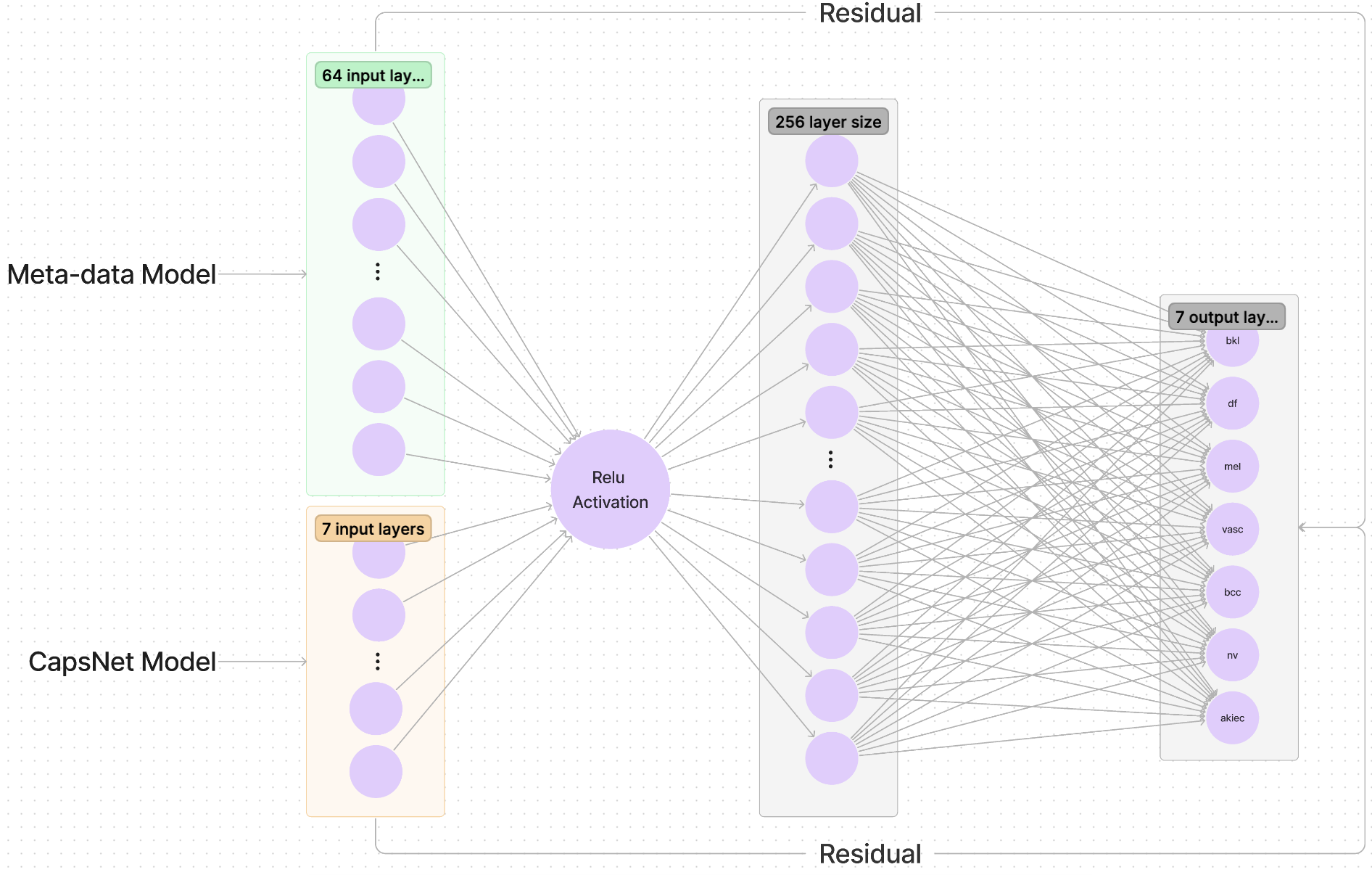
**Appendix 4: Dataset with 40,000 images class distribution**

|  | **akiec** | **bcc** | **bkl** | **df** | **mel** | **nv** | **vasc** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Train** | 5593 | 6286 | 6344 | 4701 | 6322 | 6433 | 5676 |
| **Test** | 1399 | 1572 | 1587 | 1176 | 1581 | 1609 | 1420 |

**Appendix 5: Meta-Data Model Architecture**

****

**Appendix 6: Fusion Model Architecture**

****

### Resources

1. Patrick, M. K., Adekoya, A. F., Mighty, A. A., & Edward, B. Y. (2022, January). Journal of King Saud University - Computer and Information Sciences. https://www.sciencedirect.com/science/article/pii/S1319157819309322#:~:text=In%20a%20capsule%20network%2C%20each,or%20features)%20from%20a%20CNN
2. Draelos, V. a. P. B. R., MD PhD. (2019, September 15). Best Use of Train/Val/Test Splits, with Tips for Medical Data. Retrieved from <https://glassboxmedicine.com/2019/09/15/best-use-of-train-val-test-splits-with-tips-for-medical-data/>
3. Lan, Z., Cai, S., He, X., & Wen, X. (2022, June 8). FixCaps: An Improved Capsules Network for Diagnosis of Skin Cancer. <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=9791221>
4. Mujahid, M., Rustam, F., Álvarez, R. M., Mazón, J. L. V., De La Torre Díez, I., & Ashraf, I. (2022). Pneumonia Classification from X-ray Images with Inception-V3 and Convolutional Neural Network. Diagnostics, 12(5), 1280. <https://doi.org/10.3390/diagnostics12051280>
5. Yang, G., Luo, S., & Greer, P. (2023, March 27). A Novel Vision Transformer Model for Skin Cancer Classification. https://link.springer.com/article/10.1007/s11063-023-11204-5
6. Ali, K., Shaikh, Z. A., Khan, A. A., & Laghari, A. A. (2022). Multiclass skin cancer classification using EfficientNets – a first step towards preventing skin cancer. Neuroscience Informatics, 2(4), 100034. <https://doi.org/10.1016/j.neuri.2021.100034>
7. Gupta, A. (2024, January 23). *A comprehensive guide on Optimizers in deep learning*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-deep-learning-optimizers/
8. Loshchilov, I., & Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.
9. Almeida, V. (2023, May 2). *Grad-cam in pytorch: Use of forward and Backward Hooks*. Medium. <https://towardsdatascience.com/grad-cam-in-pytorch-use-of-forward-and-backward-hooks-7eba5e38d569>
10. HAM10000(aug)\_数据集-飞桨AI Studio星河社区. (n.d.). Retrieved from <https://aistudio.baidu.com/datasetdetail/151696>