



LUT School of Energy Systems

Bachelor's thesis, Electrical engineering

**Evaluating Nvidia Jetson Nano 2GB for CNN Image Classification**

**Nvidia Jetson Nano 2GB:n arviointi CNN-pohjaiseen kuvien luokitteluun**

29.5.2025

Author: Niko Laurén

Supervisor: Dick Carrillo Melgarejo

## ABSTRACT

<b>Author</b>	Niko Laurén
<b>Title</b>	Evaluating Jetson Nano 2GB for CNN Image Classification
<b>School</b>	LUT School of Energy Systems
<b>Degree programme</b>	Electrical engineering
<b>Supervisor</b>	Dick Carrillo Melgarejo
<b>Keywords</b>	Nvidia Jetson Nano, Convolutional Neural Networks, Image classification, ResNet-18, Model retraining

This thesis focuses on evaluating the capabilities of the Nvidia Jetson Nano 2GB Developer Kit for image classification tasks using Convolutional Neural Network (CNN) models. In this study, the classification accuracy of pretrained CNN models is assessed. The study also focuses on comparing the performance of the pretrained ResNet-18 model with a retrained model on the Jetson Nano platform.

Two datasets are used to retrain the ResNet-18 model: a new custom dataset consisting of images of coffee mugs and the Cats & Dogs dataset provided by Nvidia. The impact of dataset size and diversity on classification accuracy was also investigated.

The evaluation consists of measuring the accuracies of each pretrained model for different classes with images provided by Nvidia as well as the images collected from the internet. Unexpected results arose during retraining the ResNet-18 with limited datasets, including a decrease in classification accuracy for cats. The CNN models seem to perform much better with input images that resemble those in the training datasets. The results of retraining indicate that the dataset characteristics, such as size and diversity of images significantly affect model performance, at least on the Jetson Nano.

## TIIVISTELMÄ

<b>Tekijä</b>	Niko Laurén
<b>Tutkielman nimi</b>	Jetson Nano 2GB:n arviointi CNN-pohjaiseen kuvien luokitteluun
<b>Akateeminen yksikkö</b>	LUT School of Energy Systems
<b>Koulutusohjelma</b>	Sähkötekniikka
<b>Ohjaaja</b>	Dick Carrillo Melgarejo
<b>Hakusanat</b>	Nvidia Jetson Nano, Konvoluutioneuroverkot Kuvien luokittelu, ResNet-18, Mallien uudelleenkoulutus

Tämä opinnäytetyö keskittyy Nvidia Jetson Nano 2GB Developer Kitin suorituskyvyn arviointiin kuvien luokittelutehtävissä CNN-malleja käyttäen. Tässä tutkimuksessa arvioidaan esikoulutettujen CNN-mallien luokittelutarkkuutta. Tutkimus keskittyy myös esikoulutetun ResNet-18 -mallin suorituskyvyn vertaamiseen Jetson Nano -alustalla uudelleen koulutettuun mallin kanssa.

ResNet-18-mallin uudelleenkouluttamiseen käytetään kahta tietoaaineistoa: uutta, räätälöityä kahvimukeista koostuvaa tietoaaineistoa, sekä Cats & Dogs -tietoaaineistoa, jonka Nvidia tarjoaa. Työssä tutkittiin myös tietoaaineiston laajuuden ja monimuotoisuuden vaikutusta luokittelun tarkkuuteen.

Arviointi koostuu kunkin esikoulutetun mallin tarkkuuden mittaamisesta eri luokilla sekä Nvidian tarjoamilla kuvilla että internetistä kerätyillä kuvilla. Odottamattomia tuloksia syntyi ResNet-18 -mallin uudelleenkoulutuksen aikana rajoitetuilla tietoaaineistoilla, mukaan lukien kissojen luokittelutarkkuuden heikkeneminen. CNN-mallit näyttävät toimivan paljon paremmin kuvilla, jotka muistuttavat koulutustietoaaineistossa olevia kuvia. Uudelleenkoulutuksen tulokset osoittavat, että tietoaaineiston ominaisuudet, kuten kuvien koko ja monimuotoisuus, vaikuttavat merkittävästi mallien suorituskykyyn, ainakin Jetson Nanolla.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Background</b>	<b>2</b>
<b>3</b>	<b>Basic usage of 2GB-Jetson Nano</b>	<b>3</b>
3.1	Overview of JetPack and L4T . . . . .	3
3.2	Docker Container . . . . .	3
3.3	Running applications . . . . .	3
3.4	Verifying Pytorch . . . . .	5
3.5	Retraining models . . . . .	6
<b>4</b>	<b>Testing and Retraining</b>	<b>7</b>
4.1	Testing pretrained models with images . . . . .	7
4.1.1	Testing with example images . . . . .	7
4.1.2	Testing with custom images . . . . .	8
4.2	Retraining ResNet-18 . . . . .	8
4.2.1	Preparation for retraining . . . . .	8
4.2.2	Retraining with Nvidia’s dataset . . . . .	9
4.2.3	Creating own dataset . . . . .	10
4.2.4	Retraining with own dataset . . . . .	12
<b>5</b>	<b>Results</b>	<b>14</b>
5.1	Inference results for pretrained models . . . . .	14
5.1.1	Images provided by Nvidia . . . . .	14
5.1.2	Images obtained from the internet . . . . .	15
5.1.3	Comparison of average accuracies . . . . .	16
5.2	Inference results of retraining with Cats & Dogs . . . . .	17
5.3	Inference results of retraining with Coffee mugs . . . . .	18
<b>6</b>	<b>Discussion</b>	<b>19</b>
6.1	Inference performance of the network models . . . . .	19
6.1.1	Performance with images provided by Nvidia . . . . .	19

6.1.2	Performance with images from the internet . . . . .	19
6.2	Inference with Cats & Dogs -dataset . . . . .	19
6.3	Inference with coffee mug -dataset . . . . .	19
6.4	Summary . . . . .	20
<b>7</b>	<b>Conclusions</b>	<b>21</b>

## Nomenclature

<b>AI</b>	Artificial Intelligence
<b>CLI</b>	Command Line Interface
<b>CNN</b>	Convolutional Neural Network
<b>CUDA</b>	Compute Unified Device Architecture
<b>CSI</b>	Camera Serial Interface
<b>Enum</b>	Enumeration Type
<b>GB</b>	Gigabyte
<b>GPU</b>	Graphics Processing Unit
<b>GUI</b>	Graphical User Interface
<b>JPEG</b>	Joint Photographic Experts Group
<b>L4T</b>	Linux For Tegra
<b>MB</b>	Megabyte
<b>MIPI</b>	Mobile Industry Processor Interface
<b>ML</b>	Machine Learning
<b>NN</b>	Neural Network
<b>ONNX</b>	Open Neural Network Exchange
<b>OS</b>	Operating System
<b>RAM</b>	Random Access Memory
<b>SDK</b>	Software Development Kit
<b>ZRAM</b>	Compressed Random Access Memory Swap

# 1 Introduction

The objective of this Bachelor's thesis is to investigate the suitability of the Jetson Nano 2 Gigabyte (GB) Developer Kit for implementing Artificial Intelligence (AI) applications, such as image classification, and to provide a comprehensive analysis to determine whether Jetson Nano is the suitable platform for training CNN models like ResNet-18. The work focuses on testing the capabilities of the Jetson Nano, assessing its strengths and limitations in training, and comparing the performance of different network models.

This research topic was chosen because it is necessary to assess how well this platform can perform cognitive tasks in a resource-constrained embedded environment. Jetson Nano is an affordable, efficient and energy-saving minicomputer introduced by Nvidia, specifically designed for smart applications. It supports machine learning libraries such as TensorFlow and PyTorch and provides real-time computing capabilities in a compact form factor. In this thesis, the performance of the Jetson Nano is evaluated practically by a hands-on testing with pretrained CNN models as well as retraining a model with a preexisting and a custom dataset to determine its potential for a lightweight AI use cases.

In previous research Süzen et al. (2020), the efficiency and performance of Jetson Nano, Raspberry Pi 4, and Jetson TX2 were analyzed in the development of learning algorithms. The study found that Jetson Nano achieved higher accuracy, especially with smaller datasets, while maintaining low costs and power consumption. Jetson Nano is therefore an attractive option for training AI models, particularly with a limited budget.

Research Questions:

- (1) How suitable is the Jetson Nano 2GB Developer Kit for implementing image classification tasks using CNN models such as ResNet-18?
- (2) What are the key performance differences between pretrained CNN models and a retrained ResNet-18 model on the Jetson Nano?
- (3) What impact does the size and diversity of a dataset have on the classification accuracy on the Jetson Nano platform?

These research questions aim to evaluate the applicability of the Jetson Nano for image classification tasks. The first question examines how well the Jetson Nano, which operates with limited computational resources, performs when running CNN models. In this context, the classification accuracy is compared between all the pretrained models provided by Nvidia. The second question compares the classification accuracy of a pretrained ResNet-18 model with the same model retrained using new datasets. The third question addresses the impact of dataset size and diversity on the classification accuracy of the Jetson Nano, especially in situations where only limited datasets are available.

## 2 Background

AI and Machine Learning (ML) have become more common in the recent years for various applications including healthcare Shaheen (2021), industries Rashid and Kausik (2024) and autonomous vehicles Bathla et al. (2022). This has increased the demand for AI inference and solutions that are affordable and compact. One possible candidate is the Nvidia Jetson Nano, a small, energy efficient AI computer Nvidia (2025a) designed to handle AI workloads like image classification and object detection with great performance. This chapter briefly goes through the theoretical concepts relevant to this thesis and to train the Jetson Nano. This lays the foundation for the practical implementation of the thesis.

Inference (also known as network execution or forward pass) is referring to the phase where a already trained Neural Network (NN) processes new input data, in this case images to generate predictions. Data is passed through the network layers producing a probability (confidence score) that indicates the likelihood of the input image belonging to a predefined class Velasco-Montero et al. (2022).

CNN is a widely used architecture type in AI models for image classification, object recognition and facial recognition. CNNs are designed to learn relevant patterns from the input images by passing them through multiple convolutional layers. Each layer has filters for extracting increasingly abstract features from the images Taye (2023).

ML refers to the computers ability to improve its performance by learning from experience represented by data. Like humans, computer can make decisions and predictions based on the past experiences by analyzing large amounts of data. A ML algorithm builds a model from the data making the system able to generalize and apply what has been learned to future inputs. With ML, the intelligent systems become more effective Zhou (2021).

Image classification is a process where the neural network algorithm assigns a label to an image based on the object that the image is representing. Usually the image contains one object and a background. This task present challenges to the computer since the object classes are defined by human perception rather than consistent visual features. This results to large variations between the images of the same class Toennies (2024).

Understanding the fundamental concepts of training the CNN models provides context for the practical implementation of this thesis. The next chapters go through the process of setting up the Jetson Nano and training the network models with it.

## 3 Basic usage of 2GB-Jetson Nano

This section will explain the basics of setting up the Jetson Nano with an imagenet program using Nvidia (2023a) GitHub guide as a reference. There are two primary ways to configure the Jetson Nano after flashing it with an SD card image Nvidia (2025b). The two methods are to build the project from a source or using a docker container. DockerHub hosts the docker container images that come prebuilt for this project. Using a container makes the setup process more straightforward, as there is no need to manually build the project.

### 3.1 Overview of JetPack and L4T

Nvidia JetPack is an Software Development Kit (SDK) designed for Jetson devices that provides a solution for developing and running ML and AI applications. JetPack enables easy installation of the Operating System (OS) and drivers while including essential libraries such as the Linux For Tegra (L4T) Linux kernel, Compute Unified Device Architecture (CUDA) Toolkit, cuDNN, TensorRT, and more Nvidia (2023b). L4T is a board support package developed for Jetson Nano, which includes, among other things, the Linux kernel and bootloader.

### 3.2 Docker Container

Docker Container images are available for different JetPack versions. The appropriate container tag must match the installed JetPack-L4T version on the Jetson Nano. To start using the container, download the `jetson-inference` repository with `docker pull dustynv/jetson-inference`: command followed with the correct L4T version. For JetPack 5.0.2 the L4T version is 35.1.0 Nvidia (2024). Navigate to the newly created `jetson-inference` directory and run the container in the terminal using the provided script `run.sh`.

```
$ docker pull dustynv/jetson-inference:r35.1.0
$ cd jetson-inference
$ docker/run.sh
```

All necessary mounts and devices are configured to enable camera and display functionality within the container. The `docker/run.sh` command runs the container and the keyboard shortcut `Ctrl + D` exits the container Nvidia (2024).

### 3.3 Running applications

When running the container, the example programs can be executed by navigating to the `/bin` -directory with the `cd build/aarch64/bin` command. In the correct directory type, for example,

```
./imagenet.py images/jellyfish.jpg images/test/jellyfish.jpg
```

This command runs the ImageNet program, which classifies an example image `jellyfish.jpg` and saves the result to the `images/test/` -directory with a confidence score in the top left corner of the image (figure 1) Nvidia (2023a). ImageNet uses GoogleNet as the default classification model. To use a different classification model, a `--network` flag needs to be used. For example:

```
./imagenet.py --network=resnet-18 images/jellyfish.jpg images/test/jellyfish.jpg
```

Where `resnet-18` is the corresponding Command Line Interface (CLI) argument of the ResNet-18 model. The available pre-trained classification models can be found in the table 1.

Network	CLI argument	NetworkType Enumeration Type (Enum)
AlexNet	alexnet	ALEXNET
GoogleNet	googlenet	GOOGLENET
GoogleNet-12	googlenet-12	GOOGLENET_12
ResNet-18	resnet-18	RESNET_18
ResNet-50	resnet-50	RESNET_50
ResNet-101	resnet-101	RESNET_101
ResNet-152	resnet-152	RESNET_152
VGG-16	vgg-16	VGG-16
VGG-19	vgg-19	VGG-19
Inception-v4	inception-v4	INCEPTION_V4

Table 1: Classification models, corresponding CLI arguments and NetworkType Enum Nvidia (2023a)



Figure 1: Example inference output of a jellyfish Nvidia (2023a).

### 3.4 Verifying Pytorch

Before retraining any models, it is necessary to ensure that PyTorch is installed correctly and the desired Graphics Processing Unit (GPU) is detected. This can be accomplished by running Python shell in the terminal with `python` or `python3`-command and executing the print-commands for the torch version and CUDA availability. To verify the version of torch and torchvision, the libraries can be imported into the python shell and their version can be printed out Nvidia (2023c).

```
>>> import torch
>>> print(torch.__version__)
>>> import torchvision
>>> print(torchvision.__version__)
```

To verify the detection of the GPU and its tensors, the `torch.cuda.is_available()` function from the torch-library can be used.

```
>>> print('CUDA available: ' + str(torch.cuda.is_available()))
>>> a = torch.cuda.FloatTensor(2).zero_()
>>> print('Tensor a = ' + str(a))
>>> b = torch.randn(2).cuda()
>>> print('Tensor b = ' + str(b))
>>> c = a + b
>>> print('Tensor c = ' + str(c))
```

### 3.5 Retraining models

Retraining pre-trained classification models using transfer learning is generally much faster than a training model from the ground up Nvidia (2023c). Re-training models with transfer learning on the Jetson Nano has been made easy. The following example is used to demonstrate training the ResNet-18 model with the Cat/Dog dataset.

First, to download the dataset, the `wget` command is used to retrieve the dataset files from the web. The files are downloaded from the specified web link. The files are saved in the `tar.gz` format. After downloading, the `tar` command is used to extract the downloaded files. The arguments `xvzf` of the `tar` command define, how the extraction should be done Iftikhar (2025).

- `x`: Used to extract the files
- `v`: Show the progress
- `z`: Unzips the `tar.gz` files
- `f`: Tells that the next part of the command is a filename

```
$ cd jetson-inference/python/training/classification/data
$ wget https://nvidia.box.com/shared/static/
o577zd8yp3lmxf5zhm38svrbrv45am3y.gz -O cat_dog.tar.gz
$ tar xvzf cat_dog.tar.gz
```

To actually retrain a model, in the `jetsoninference/python/training/classification` directory, run a python script `train.py`. This will start the training process. Training can be stopped any time by pressing `Ctrl+C` and continued with `--resume` Nvidia (2023d).

```
$ python3 train.py --model-dir=models/cat_dog data/cat_dog
```

After retraining has been successfully completed, the PyTorch model needs to be converted to Open Neural Network Exchange (ONNX):

```
$ python3 onnx_export.py --model-dir=models/cat_dog
```

## 4 Testing and Retraining

This section describes the practical implementation of the thesis. The implementation is divided into subsections including testing the pretrained models and retraining the ResNet-18 model with datasets to create new models.

### 4.1 Testing pretrained models with images

The first step of the implementation is to test all the available pre-trained image classification models provided by Nvidia. GoogleNet-12 was excluded from this testing due to errors encountered during inference.

#### 4.1.1 Testing with example images

The models listed in the table 1 were tested with 10 example pictures located in the docker container. These images represented various object categories, including cats and dogs, fruits, a coffee mug and other animals. Inference was done for each image with every available model resulting in a total of 90 confidence scores. The purpose of this test was to evaluate the classification accuracy and to establish a performance base line of each model.

The results of this test are found in the table 3, that presents the classification accuracies (%) by each model for each image. The images in the container are used to train the models, so they had great accuracies only tenths of a percentile away from a 100% and some of them even had perfect scores as seen in the figure 2 representing a strawberry.



Figure 2: Output of the inference with GoogleNet for a strawberry (100.00%).

### 4.1.2 Testing with custom images

To get a better understanding of the performance of these models, more images were acquired from the internet. New set of 10 images representing the same classes were collected and tested with each model again. The results for the classification accuracies are found in the table 4.

Some images were very similar to the ones provided by Nvidia and the classification models performed as well as with the previous images, however other images, that were somewhat different from the images used to train the models had worse accuracy scores as low as 12% or even misses where the model fails to classify the image correctly and believes that it's representing a totally different class like in the figure 3.

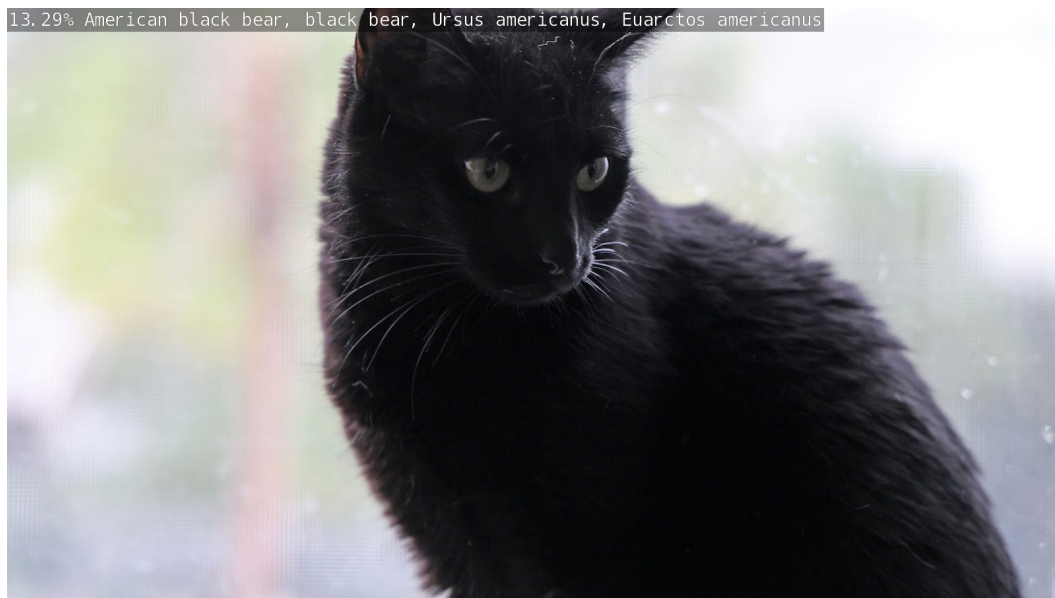


Figure 3: Output of the inference with ResNet-18 for a cat (Black bear 13.29%).

## 4.2 Retraining ResNet-18

This section explains the methods used for retraining the ResNet-18 model with a Cats & Dogs dataset provided by Nvidia and a custom dataset consisting of coffee mugs collected for this thesis. This section also goes through the necessary preparations to make sure that the retraining is efficient.

### 4.2.1 Preparation for retraining

Before retraining the model with any dataset, some preparation can be made to ensure efficient and smooth training. First it is necessary to ensure, that the PyTorch is installed correctly and the desired GPU is detected. This can be done like in the section 3.4.

Since retraining models uses a lot of memory, a mounting swap is recommended Nvidia (2023c). Mounting swap refers to the process of using parts of the storage memory as an extension for the system's physical Random Access Memory (RAM).

Portions of the storage memory hold the inactive memory pages temporarily to free up the space on the actual RAM. This process makes the system manage memory efficiently when running applications that require lots of memory Community (2022).

The basic principle of mounting swap is the same in the Jetson Nano as in any other Linux-based computer. First `sudo systemctl disable nvzramconfig` can be used to disable Compressed Random Access Memory Swap (ZRAM) configuration process. To create a four GB file for the swap, `sudo fallocate -l 4G /mnt/4GB.swap` is used. Then, `sudo mkswap /mnt/4GB.swap` and `sudo swapon /mnt/4GB.swap` are used to allocate the file for a swap and put to use Nvidia (2023c). These changes can be made persistent across reboots by adding `/mnt/4GB.swap none swap sw 0 0` at the end of `etc/fstab` file.

```
sudo systemctl disable nvzramconfig
sudo fallocate -l 4G /mnt/4GB.swap
sudo mkswap /mnt/4GB.swap
sudo swapon /mnt/4GB.swap
```

Disabling the desktop Graphical User Interface (GUI) is another way to free up some memory for the training. The desktop GUI can be disabled temporarily with `sudo init 3` and rebooted with `sudo init 5`. To make the system boot to either with desktop GUI or without it, the following commands can be used.

```
$ sudo systemctl set-default multi-user.target # disable desktop on boot
$ sudo systemctl set-default graphical.target # enable desktop on boot
```

#### 4.2.2 Retraining with Nvidia’s dataset

Nvidia provides three datasets in the Nvidia (2023c) guide. Two of them are for the image classification and one for object detection. For image classification, there is a smaller Cat/Dog -dataset consisting of cats and dogs and a bit larger PlantCLEF -dataset consisting of fruits. Size of the Cat/Dog -dataset is around 800 Megabyte (MB) and the PlantCLEF is around 1.5 GB. For this study, the Cat/Dog -dataset was chosen for its, smaller size, training times and it’s simplicity with only two classes, one for cats and one for dogs. The datasets and their training times can be found in the table 2. Time per Epoch means the approximate time for one training cycle to complete and Training time is the approximate time that it takes for the model to be trained 35 times or Epochs with Jetson Nano.

Type	Dataset	Size	Classes	Training Images	Time per Epoch	Training Time
Classification	Cat/Dog	800MB	2	5000	7-8 min	4 hours
Classification	PlantCLEF	1.5GB	2	10475	15 min	8 hours
Detection	Fruit	2GB	2	6375	15 min	8 hours

Table 2: Datasets and their training times provided by Nvidia

The retraining with pre-existing dataset is done as described in the section Retraining models 3.5. The results of this retraining with the Cat/Dog -dataset can be found from the table 5.

### 4.2.3 Creating own dataset

First, to create a custom dataset, lots of images are needed. Collecting a large amount of images can be time consuming. Fortunately, Nvidia offers a user-friendly tool named `camera-capture` that captures and labels images. The tool creates the appropriate structure for the dataset consisting of `train`, `validation` and `test` folders. Each folder has subdirectories for each image class. These subdirectories are automatically filled by the tool when taking sequences of images Nvidia (2023e).

Example structure of the dataset created with `camera-capture`:

```
train/
  class-A/
  class-B/
  ...
val/
  class-A/
  class-B/
  ...
test/
  class-A/
  class-B/
  ...
```

To define classes for the dataset, a label file must be created to the directory `jetson-inference/python/training/classification/data` and it's generally called `labels.txt`. The label file should contain one class label each line and it should be organized by alphabetically Nvidia (2023e).

Example structure of `labels.txt` -file:

```
background
brontosaurus
tree
triceratops
velociraptor
```

When using the default Mobile Industry Processor Interface (MIPI) Camera Serial Interface (CSI) camera, the `camera-capture` -tool can be launched with `camera-capture csi://0`. A data capture control -window (figure 4) opens up, where dataset path, class labels file, current set (train, test, validation), current class and Joint Photographic Experts Group (JPEG) quality can be defined. After defining, pictures can be captured with the spacebar or by clicking the `Capture` -button and they are automatically saved in the correct folder under the correct subdirectory class.

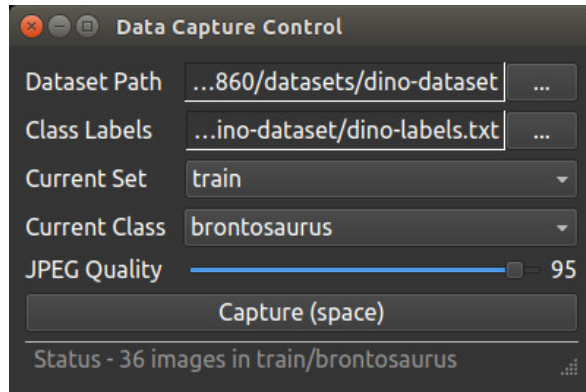


Figure 4: Data Capture Control -window Nvidia (2023c).

A minimum of 100 training images per class is recommended to ensure successful training, while the validation set should be around 10-20% of the training set Nvidia (2023e).

For this study, a coffee mug was chosen as the object, from which a custom dataset of entirely new images, taken with the `camera-capture` -tool, was created. The coffee mug -dataset consists of 501 training images and 53 validation images of four different coffee mugs with slight variation in lighting, angle (figure 5), rotation (figure 6) and layout (figure 7). The dataset consists of two classes `coffee_mug` and `other`. The `other` -class has all kinds of images from the ImageNet-database, 478 training images and 53 validation images. Due to the limitations of the stationary MIPI CSI camera, all the coffee mug images were captured at the work station in the lab.

All of the four coffee mugs were rotated on the desk and their positions were altered.



Figure 5: Coffee mug rotated on the desk

The coffee mugs were lifted from the table and held in hand in different positions and angles.



Figure 6: Coffee mug from a different angle

Close-up pictures were taken of the coffee mugs alone and with other coffee mugs.



Figure 7: Coffee mugs placed near the camera

#### 4.2.4 Retraining with own dataset

Retraining with own newly created dataset works similarly to the training of pre-existing datasets with just changing the model and the dataset to desired:

```
$ cd jetson-inference/python/training/classification
$ python3 train.py --model-dir=models/<YOUR-MODEL> data/<YOUR-DATASET>
```

After retraining the model, it needs to be converted to ONNX. Inference with the new models can be done by first defining `NET` and `DATASET` directories and then loading the model with the `Imagenet` -program.

```
NET=models/<YOUR-MODEL>
```

```
DATASET=data/<YOUR-DATASET>
```

```
imagenet.py --model=$NET/resnet18.onnx --input_blob=input_0 --  
output_blob=output_0 --labels=$DATASET/labels.txt csi://0
```

## 5 Results

This section presents the results of the inference tests performed with all the pretrained CNN models using both images that Nvidia has provided in the docker container and images that were collected from the internet. This section also covers the results from the retraining the ResNet-18 model with Cats & Dogs -dataset as well as the custom Coffee mug -dataset made for this thesis. All the results are reported as classification accuracies.

### 5.1 Inference results for pretrained models

Inference was performed for all the pretrained models provided by Nvidia. The models include AlexNet (AN), GoogLeNet (GN), various versions of ResNet (RN18, RN50, RN101, RN152), VGGNet (VGG16, VGG19), and Inception-v4 (IncV4). GoogleNet-12 was excluded due to errors encountered during testing.

#### 5.1.1 Images provided by Nvidia

Table 3 presents the inference accuracies (%) for the images found in the docker container. One of the images (Dog) was not correctly classified by any model, likely due to an issue with the image’s class label.

Image	AN	GN	RN18	RN50	RN101	RN152	VGG-16	VGG-19	IncV4
Bird	71.44	53.52	58.98	30.50	51.86	41.46	84.23	54.39	94.82
Cat	74.12	74.56	68.51	82.13	88.33	84.72	62.55	72.07	82.96
Cat 2	98.49	99.85	81.01	100	99.95	100	99.76	99.71	94.82
Dog	X	X	X	X	X	X	X	X	X
Dog 2	93.21	99.80	90.19	89.45	73.58	93.12	86.57	89.40	62.94
Orange	93.36	92.68	94.78	99.12	98.39	99.61	97.66	98.34	84.67
Strawberry	99.41	100.00	99.32	100.00	99.95	99.95	99.56	99.95	93.70
Coffee mug	49.66	72.36	70.70	89.94	51.46	93.51	91.70	85.30	78.47
Jellyfish	91.55	99.85	99.12	100.00	99.95	99.95	99.90	99.95	88.92
Polar bear	100.00	100.00	99.76	99.51	99.12	99.02	100.00	100.00	89.50

Table 3: Accuracies (%) of inference for images provided by Nvidia.

By excluding one of the tested images (Dog) from the calculations, average accuracies can be determined for each model. The results show that VGG-16 achieved the highest average accuracy at around 91.33%, second best performing model was ResNet-152 with a 90.15% accuracy. The smaller ResNet variants ResNet-18 and ResNet-101 achieved slightly lower accuracies of approximately 85.7%.

The figure 8 summarizes the average inference accuracies for each model. More complex networks tend to offer better inference accuracy, however each model is performing well. The combined average accuracy for all models was 87.44%.

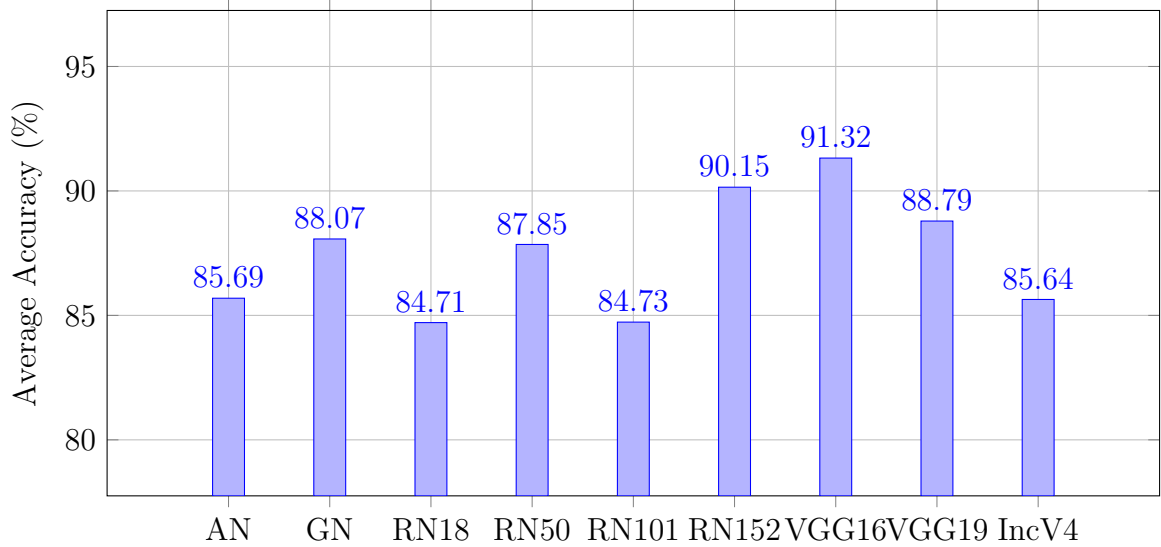


Figure 8: Average inference accuracies (%) of different models with Nvidia-provided images

### 5.1.2 Images obtained from the internet

Table 4 presents the inference accuracies (%) for the images collected from the internet except the coffee mug that was taken with the Jetson Nano’s camera. Some of the images are very similar to those found in the container, resulting in higher scores while others differ significantly. Some of the accuracies are low (20% or lower) and this shows that the classification models are only as good as the datasets they are trained on. In here, the failed classifications are more frequent especially with the coffee mug and the jellyfish. In addition, models ResNet-18, ResNet-50 and AlexNet failed to classify one of the cat images.

Image	AN	GN	RN18	RN50	RN101	RN152	VGG-16	VGG-19	IncV4
Bird	89.65	50.00	33.11	38.65	46.14	38.35	83.45	87.89	73.10
Cat 3	79.00	61.18	50.54	56.74	58.64	51.03	55.27	69.24	60.99
Cat 4	X	39.14	X	X	77.10	94.14	54.00	28.22	91.26
Dog 3	44.12	29.83	21.18	33.42	17.22	28.15	13.99	43.38	46.83
Dog 4	44.63	50.15	40.06	40.87	42.09	44.53	43.80	86.67	86.82
Orange	98.44	95.51	59.38	97.85	99.02	99.56	92.77	86.43	73.58
Strawberry	49.46	97.56	98.58	100.00	99.95	99.85	99.46	99.32	86.57
Coffee mug 2	12.07	59.08	X	X	X	X	X	X	77.10
Jellyfish	90.48	97.61	60.94	X	X	X	X	X	X
Polar bear	99.85	100.00	97.02	100.00	99.76	99.95	99.95	99.56	84.47

Table 4: Accuracies (%) of inference for images from the internet

Average accuracies were calculated for each model now treating failed classification (X) as a zero (0%). These accuracies are found in figure 9. Each model saw a decrease in the accuracy, but they all similar performance, when compared to each other. The combined average accuracy with internet pictures for all models was 57.06%.

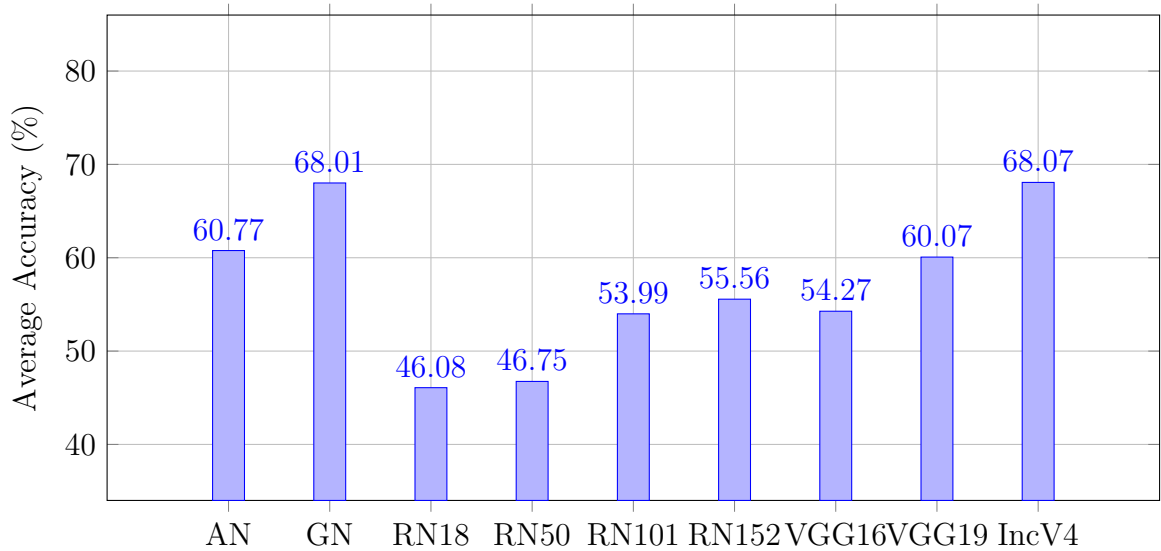


Figure 9: Average inference accuracies (%) with internet-sourced images

### 5.1.3 Comparison of average accuracies

Table 10 presents a side by side comparison of the average accuracies (%) for each model with Nvidia’s images and images from the internet. ResNet-50 experienced the largest drop in accuracy by 41.10%, while the Inception-V4 performed the best with a 17.57% decrease. Average drop in accuracy was 30.38%.

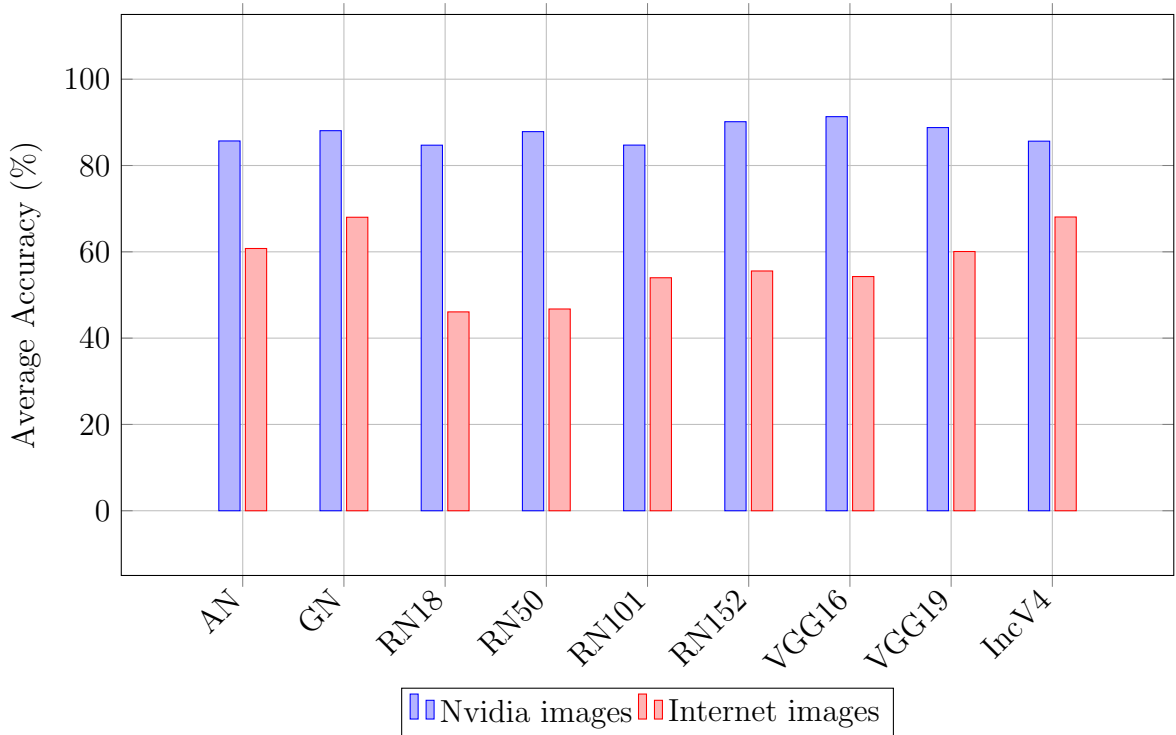


Figure 10: Comparison of average inference accuracies (%) between Nvidia and internet images for each model

## 5.2 Inference results of retraining with Cats & Dogs

After retraining the ResNet-18 model with the Cats & Dogs -dataset, the accuracies for cats got worse. The accuracies for Dogs got better except for Dog 2. While the accuracies are somewhat worse, they are much closer together which may indicate of the limitation of the dataset. These images were the same ones used previously in the inference with other objects.

Image	ResNet-18	Retrained Cats and Dogs
Cat	68.51	63.98
Cat 2	81.01	62.45
Cat 3	50.54	X
Cat 4	X	X
Dog	X	81.34
Dog 2	90.19	71.71
Dog 3	21.18	88.12
Dog 4	40.06	71.37

Table 5: Comparison of inference accuracies (%) after retraining the ResNet-18 with Cats and Dogs -dataset

### 5.3 Inference results of retraining with Coffee mugs

After training the ResNet-18 model with the custom coffee mug -dataset, the classification failed with all the other tested coffee mug images as seen in the table 6 except for the coffee mug 2 that was the one inference picture taken with the Jetson Nano’s camera. With this limited data can be concluded that the dataset was working for a very specific kind of images of coffee mugs that are similar to the images that the model was trained on and it can’t classify any other coffee mugs in a different lighting or scenery.

Image	ResNet-18	Trained own dataset
Coffee mug	70.70	X
Coffee mug 2	X	90.24
Coffee mug 3	34.20	X
Coffee mug 4	39.09	X
Coffee mug 5	10.86	X

Table 6: Comparison of inference accuracies (%) after retraining the ResNet-18 with own dataset

## 6 Discussion

The results presented in the previous section provided some interesting information about the strengths and limitations of the CNN models. These models perform well when the datasets they are trained on are sufficiently diverse.

### 6.1 Inference performance of the network models

The inference results of the images provided by Nvidia (3) and the images collected from the internet (4) were good and bad at the same time. On the other hand, these models perform well with images that are well-aligned with the training data, but when the images have more variance and deviation from the training dataset, it seems that the models are performing worse.

#### 6.1.1 Performance with images provided by Nvidia

Average accuracy for the images, that came with the docker container, was great (85.7%). The excluded dog image would have lowered the accuracies. Another factor that should be taken into consideration is that the amount of pictures that were inferred was limited. If more inferences were performed, the average accuracy might be slightly lower.

#### 6.1.2 Performance with images from the internet

For the images, that were collected from the internet and the one coffee mug -image taken with the Jetson Nano's camera, the average accuracy was worse (57.06%). This was partly due to the missed classifications (X) that were handled as a 0%. Even without the missed classifications, all the accuracies, especially with the smaller models, had a decrease. Only images like the polar bear and orange hold the classification accuracy due to the similarity of these images compared to the ones provided by Nvidia. One reason to the differences between the decrease of the models accuracies might be that some of the models had datasets consisting of images more similar to the ones found from the internet.

### 6.2 Inference with Cats & Dogs -dataset

Inference results after training the ResNet-18 -model with the Cats & Dogs -dataset (5) were surprising. For some reason, the accuracies for the cat images went down. There might be some error with the limited amount of inferences made, but also the smaller size of the dataset probably has an effect on the accuracies. The accuracies for the dog images had an increase, but Dog 2 accuracy decreased from 90.19% down to 71.71%.

### 6.3 Inference with coffee mug -dataset

The ResNet-18 -model learned to recognize the coffee mug (6) used in the training dataset. However, it didn't recognize any of the other coffee mugs that were inferred.

This was most likely due to the homogeneity of the coffee mug images and that the **other** class images were taken from the ImageNet -database and were diverse compared to the images taken with the Jetson Nano. The model appeared to have learned to recognize the background rather than the coffee mugs themselves. That's why every time the image had even slightly different lighting or scenery, it was automatically classified as other than a coffee mug. Possible solution to this could have been adding images of the coffee mug scenery without the mugs to the **other** class to make the model recognize the difference between the background and the object.

## 6.4 Summary

The inference results indicate that the CNN models perform well when the images are similar to their training sets, but their accuracy drops when the the images are more varied. Retraining with a limited dataset can improve the recognition of the specific images but at the same time it might reduce the general recognition ability. Therefore the datasets used to train the models play a critical role when trying to achieve reliable inference performance.

## 7 Conclusions

The objective of this thesis was to investigate the suitability of the Jetson Nano 2GB Developer Kit computer for AI applications by testing pretrained image classification models and retraining the ResNet-18 model with both Nvidia’s pre-existing dataset and a custom dataset. Based on the results, multiple conclusions can be made regarding the limitations, strengths and potential applications of the Jetson Nano.

The general inference results suggest that there are some differences between the pretrained models. Models, such as ResNet-152 and VGG-16, provided the highest accuracies with the Nvidia’s images and the Inception-V4 and the GoogleNet performed well with the images obtained from the internet. Simpler models, such as ResNet-18, performed less effectively. Especially, Inception-V4 was a surprising performer with the internet images, suggesting that those images were more similar to the ones used to train Inception-V4.

Retraining with the Cats & Dogs dataset produced unexpected results. The classification accuracy for cats decreased, while the accuracy for dogs improved to some extent except for the image Dog 2, for which the accuracy also decreased. The unexpected results may be due to overfitting on the limited dataset, which leads to a loss of generalization capability.

From the inference results related to coffee mugs, it can be concluded that the Jetson Nano was able to effectively train the ResNet-18 model. However, a limitation in this study was the MIPI CSI camera, which was directly connected to the computer. As a result, the dataset became too homogeneous, and despite efforts to diversify it, the images ended up being quite similar to each other. Consequently, the model learned to recognize only a specific type of coffee mug in limited conditions and failed to generalize to other kinds of images.

Based on the Cats & Dogs dataset and the coffee mug experiment, it can be concluded that the CNN models need larger, even over 10 000-image datasets, with diverse images to classify them correctly and with high (e.g., over 90%) confidence scores. It can be argued that the limited processing capabilities of the Jetson Nano is not sufficient to train models using datasets large enough so that the models could recognize all the images in all scenarios. However, Jetson Nano can be used for training models, that only need to classify images for example from a similar environments.

Further research is needed to comprehensively evaluate the Jetson Nano’s capabilities in AI applications especially in image classification. One of the key limitations of this study was the small and homogeneous dataset in the coffee mug experiment. In the future, larger and more heterogeneous datasets should be utilized to better assess the performance of models trained on the Jetson Nano. Future studies could explore the use of transfer learning to improve the accuracy of the CNN models on image classification with the Jetson Nano for real-world applications.

## References

- A. A. Süzen, B. Duman, and B. Şen. Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn. *International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–4, 2020. doi: 10.1109/HORA49412.2020.9152915.
- Mohammed Yousef Shaheen. Scienceopen preprints. ScienceOpen, September 2021. URL <https://www.scienceopen.com/document?vid=a985f55b-7ee7-4f37-bcc3-7984e9c861ac>. Preprint, DOI: 10.14293/S2199-1006.1.SOR-.PPVRY8K.v1.
- Adib Bin Rashid and MD Ashfakul Karim Kausik. Ai revolutionizing industries worldwide: A comprehensive overview of its diverse applications. *Hybrid Advances*, 7:100277, 2024. ISSN 2773-207X. doi: <https://doi.org/10.1016/j.hybadv.2024.100277>. URL <https://www.sciencedirect.com/science/article/pii/S2773207X24001386>.
- Gourav Bathla, Kishor Bhadane, Rahul Kumar Singh, Rajneesh Kumar, Rajanikanth Aluvalu, Rajalakshmi Krishnamurthi, Adarsh Kumar, R. N Thakur, and Shakila Basheer. Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. *Mobile Information Systems*, 2022(1):7632892, 2022. doi: <https://doi.org/10.1155/2022/7632892>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1155/2022/7632892>.
- Nvidia. Robotics and edge ai, 2025a. URL <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/product-development/>. Accessed: 2025-05-21.
- Delia Velasco-Montero, Jorge Fernández-Berni, and Ángel Rodríguez-Vázquez. *Visual inference for IOT systems: a practical approach*. Springer, Cham, Switzerland, 2022. ISBN 978-3-030-90903-1. URL <https://link.springer.com/book/10.1007/978-3-030-90903-1>. Accessed: 2025-05-21.
- Mohammad Mustafa Taye. Theoretical understanding of convolutional neural network: Concepts, architectures, applications, future directions. *Computation*, 11(3), 2023. ISSN 2079-3197. doi: 10.3390/computation11030052. URL <https://www.mdpi.com/2079-3197/11/3/52>.
- Zhi-Hua Zhou. *Machine Learning*. Springer, Gateway East, Singapore, 2021. Print.
- Klaus D. Toennies. *An Introduction to Image Classification: From Designed Models to End-to-End Learning*. Springer Nature Singapore, Singapore, 1 edition, 2024. URL <https://link.springer.com/book/10.1007/978-981-99-8123-4>. Online resource.
- Nvidia. Hello ai world, 2023a. URL <https://github.com/dusty-nv/jetson-inference/blob/master/docs/imagenet-console-2.md>. Accessed: 2025-17-3.

- Nvidia. Getting started with jetson nano 2gb developer kit, 2025b. URL <https://developer.nvidia.com/embedded/learn/get-started-jetson-nano-devkit#intro>. Accessed: 2025-24-3.
- Nvidia. Jetpack setup, 2023b. URL <https://github.com/dusty-nv/jetson-inference/blob/master/docs/jetpack-setup-2.md>. Accessed: 2025-24-3.
- Nvidia. Running the docker container, 2024. URL <https://github.com/dusty-nv/jetson-inference/blob/master/docs/aux-docker.md>. Accessed: 2025-24-3.
- Nvidia. Transfer learning with pytorch, 2023c. URL <https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-transfer-learning.md>. Accessed: 2025-10-4.
- Hasib Iftikhar. Linux tar command: Everything you need to know, 2025. URL <https://cyberpanel.net/blog/linux-tar-command>. Accessed: 2025-10-4.
- Nvidia. Re-training on the cat/dog dataset, 2023d. URL <https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-cat-dog.md>. Accessed: 2025-10-4.
- Ubuntu Community. Swapfaq, 2022. URL <https://help.ubuntu.com/community/SwapFaq>. Accessed: 2025-19-5.
- Nvidia. Collecting your own classification datasets, 2023e. URL <https://github.com/dusty-nv/jetson-inference/blob/master/docs/pytorch-collect.md>. Accessed: 2025-19-5.