

Deep learning-based image segmentation of 2D NbSe₂ flakes

by

Abel de Lange

to obtain the degree of Bachelor of Science
at the Delft University of Technology,
to be defended publicly on 23-5-2025

Student Number:

Project Duration:

Faculty:

Department:

Daily Supervisors: M.P. Dubbelman
M.N. Ali

TU Delft

Deep learning-based image segmentation of 2D NbSe2 flakes

de Lange, A.L. (5607272)*

Applied Physics, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft.

(Dated: May 23, 2025)

Deep learning is a powerful tool that can be used to automate various tasks. Here, deep learning is used in the field of quantum nanoscience to predict how thick 2D material flakes are, to locate them and to create high resolution masks. Mask R-CNN, a deep learning computer vision model, trained on Graphene, MoS2, WTe2 and hBN, is fine-tuned on NbSe2 using transfer learning. To accomplish this, a dataset of optical images and AFM measurements is created. All data is gathered in a glove box at room temperature. This dataset is expanded by an image splitting algorithm. All four starting models produce an approximately equal performing NbSe2 model. With the hBN model being the best starting model. Which was not expected based on the optical properties of the materials. This might be caused by some unknown similarities between the datasets. For large flakes ($> 96px^2$) the model performs sufficiently well for implementation in automated flake searching (AP@IoU_{50%:95%} of 0.54). The model is strongly biased towards one class due to the unbalanced dataset. With a balanced dataset, this project enables researchers to build an automated flake searching setup.

I. INTRODUCTION

Two-dimensional (2D) materials are an active topic in materials research due to their interesting properties (optical, mechanical, electronic, magnetic and more) [1][2]. Graphene, for example, is the strongest material tested to date [3]. These properties can be modified and tuned by creating van der Waals heterostructures. This is done by stacking different 2D crystal layers with one-atomic-layer-precision[4]. Properties can also be modified by rotation. Graphene can be made into a superconductor by stacking two graphene layers with a 1.1 degree rotation[5]. Therefore, these materials can be used for many modern technological applications. Like imaging, energy storage and conversion and more[6]. 2D materials research consists of a very time consuming task, searching for useful flakes. Often, flakes must be in a specific size and thickness range, which depends on the type of research.

Deep learning is a branch of machine learning that uses neural networks to perform tasks such as computer vision, for example recognizing objects in images. The power of deep learning lies in the ability to learn general features, instead of having a strict set of (human written) rules. This is used from self driving cars to analyzing biomedical images[7][8].

The search of 2D flakes can for example be automated by a computer vision technique called image segmentation. This is the task of dividing an image into groups of pixels. A motorized microscope with auto focus can be instructed to scan a wafer containing flakes. During the scan, a deep learning model can make predictions. This results in the location, size and a thickness estimation of all the flakes on a wafer. Doing this overnight will significantly speed up research needing specific flakes [9].

This project aims to fine-tune the deep learning-based image segmentation model Mask R-CNN, which has been trained on 2D flakes of graphene, hexagonal boron nitride (hBN), molybdenum disulfide (MoS2) and tungsten ditelluride (WTe2), and niobium diselenide (NbSe2)[9]. This will be done by using transfer learning, using the knowledge of an existing model to create a new model. The new model will then build on the old model. [9] states that about 80 images are needed to fine-tune MRCNN on a new material, this will be the data benchmark. The resulting MRCNN model will then be trained from four transfer learning stages, shown below. See table VII in the appendix for the class and object counts of the 2D material dataset.

1. Microsoft COCO: Common Objects in Context[10],
~ 200k images
2. 2D materials: Graphene, hBN, MoS2 and WTe2,
~ 2000 images
3. Specific 2D material:
 - (i) Graphene, 862 images
 - (ii) hBN, 232 images
 - (iii) MoS2, 569 images
 - (iv) WTe2, 318 images
4. NbSe2, 80 images

It is also interesting to see which initial weights, which starting material, will result in the best NbSe2 model. The performance of transfer learning is strongly dependent on the similarity between the starting material and NbSe2 [11]. If the two materials are similar, less adaptation is needed, making training more data efficient, leading to faster convergence and potentially less overfitting [12].

Since MRCNN only receives optical images to make predictions, the similarity between materials (in this case) is determined by their optical properties. These are dependent on the band gap of a material. The band gap directly determines which wavelengths of light can be absorbed. And the color of a material is determined by which wavelengths are not absorbed. This is important

because the color of flakes is thickness dependent.

NbSe₂ is a metallic transition-metal dichalcogenide (TMDC) and a superconductor[13]. Graphene is a semimetal [14]. hBN has a large direct band gap, 6eV and a similar geometrical structure to graphene[15][16][17]. MoS₂ is a TMDC semiconductor [18] with a direct band gap of about 1.9eV[19]. And WTe₂ (a TMDC), assuming [9] used Td-WTe₂, is a semimetal [20].

Concluding, NbSe₂ has no band gap (and is a TMDC), WTe₂ also has no band gap (and also is a TMDC), and hBN has the largest band gap. Thus one could expect that training starting from WTe₂ will result in the best NbSe₂ model, and training starting from hBN will result in the worst NbSe₂ model. Also noting that hBN has been trained on the least amount of images of all initial materials.

MRCNN is trained to predict three thickness classes (bins):

- (i) Class 1: 0 - 4 layers, 0 - 2.8 nm
- (ii) Class 2: 4 - 10 layers, 2.8-7 nm
- (iii) Class 3: 10 - 40 layers, 7 - 28 nm

These are the same bins as used by [9]. The bin sizes are applicable because NbSe₂ flakes have a wide color spectrum in the thickness range of 0 to 28 nm. This results in a clear color difference between the three classes. Flakes outside this range are ignored and the model classifies them as background.

II. THEORY

A. Mask R-CNN

Mask R-CNN is a state-of-the-art computer vision model. It detects objects in images and generates segmentation masks for each instance. R-CNN stands for region-based convolutional neural network (CNN). This means that MRCNN first determines the regions of interest (RoIs) in an image, and then uses a CNN to make predictions. A CNN is a type of neural network architecture that forms the building blocks of computer vision. The main idea is that it convolves images with a specific kernel to extract features. The framework is shown below in figure 1

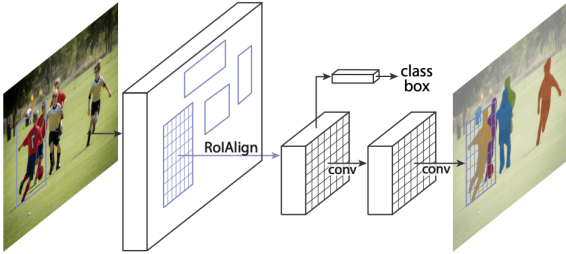


FIG. 1: The Mask R-CNN framework [21].

The output of the model consists of classification, localization, and segmentation. The localization is represented by a bounding box, a rectangle drawn in an image, defined as (x, y, w, h) , where x and y form the coordinates of the center, w is the width and h is the height. The segmentation is a binary mask, explained later on.

MRCNN consists of two differentiable sections: (i) the *backbone*, a convolutional architecture used for feature extraction, and (ii) the network *head* used for classification, localization and mask predictions for each RoI. This results in high computational costs for images with many RoIs.

1. Training

Training means updating the weights (parameters) of a model in such a way that the model improves. To know how to change the weights, a loss function is used. Essentially, a loss function states how poorly the model is performing. It is calculated using the predictions of the model and the ground truth of an image, the correct label. The multitask loss function for training is defined as[21] [22]

$$L = \alpha L_{cls} + \beta L_{box} + \gamma L_{mask} \quad (1)$$

where L_{cls} , L_{box} and L_{mask} are the classification, localization and mask losses, respectively. α , β and γ are the control parameters, used to control the influence of each loss.

The class loss, L_{cls} , is calculated from the discrete probability distribution $p = (p_0, \dots, p_K)$, over $K+1$ categories. p is computed by a softmax over the $K+1$ outputs of a fully connected layer, a layer where every neuron is connected to every other neuron. Each RoI has a corresponding ground truth class u . The log loss for true class u is given by:

$$L_{cls}(p, u) = -\log(p_u) \quad (2)$$

The localization loss, L_{box} , is calculated from the predicted bounding-box regression offsets, $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$, for each K object classes, indexed by k . t^k is an offset because it is a shift relative to the initial object proposal. For class u , the true bounding-box regression targets are defined as $v = (v_x, v_y, v_w, v_h)$. With prediction $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$. The loss is a smooth L1 loss, given by:

$$L_{bbox}(t^u, u) = \sum_{i \in \{x, y, w, h\}} \text{smooth}_{L1}(t_i^u - v_i) \quad (3)$$

where

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \quad (4)$$

This loss is less sensitive to outliers than the L2 loss, which is a parabola. This reduces the sensitivity of certain hyperparameters, such as the learning rate. Hyperparameters are the parameters that govern the training process. The learning rate determines the step size with which the model parameters are updated. The mask branch outputs K binary masks, one for each class, of size $m \times m$. A binary mask consists of two values, 1 for flake, 0 for background. Only the mask for class u is used to calculate the loss, L_{mask} . The loss is defined as the average binary cross-entropy loss.

$$L_{mask} = -\frac{1}{m^2} \sum_{1 \leq i, j \leq m} y_{ij} \log(\hat{y}_{ij}^u) + (1 - y_{ij}) \log(1 - \hat{y}_{ij}^u) \quad (5)$$

where y_{ij} is the ground truth mask at point (i, j) , and $\hat{y}_{ij}^u$ the predicted mask for class u .

In summary, MRCNN is a model that divides an image into multiple classified pixel groups. This is done by first determining all the interesting areas of the image, and then making a prediction for each interesting area.

III. METHOD

A. Data Pipeline

Data consists of two elements: (i) images, taken with zoom 100x and auto exposure and (ii) atomic force microscope (AFM) measurements. The AFM is a device that can measure the thickness of flakes with atomic layer precision. The data pipeline is shown in figure 2 (a).

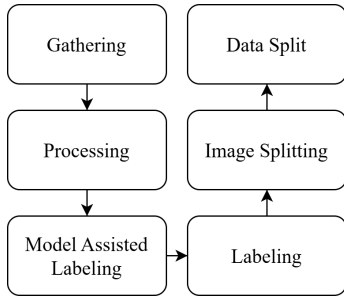


FIG. 2: Data pipeline

1. Data Gathering

Data gathering consists of three parts: (i) exfoliating NbSe2 flakes on Si/SiO2 wafers using the 'Scotch Tape' method (shown in figure 3 (b)), (ii) taking microscope images of promising flakes and (iii) performing AFM measurements to get height data of the flakes. For the dataset only images with zoom 100x are used, resulting in 80 images with AFM data. All data is collected in a glove box at room temperature.

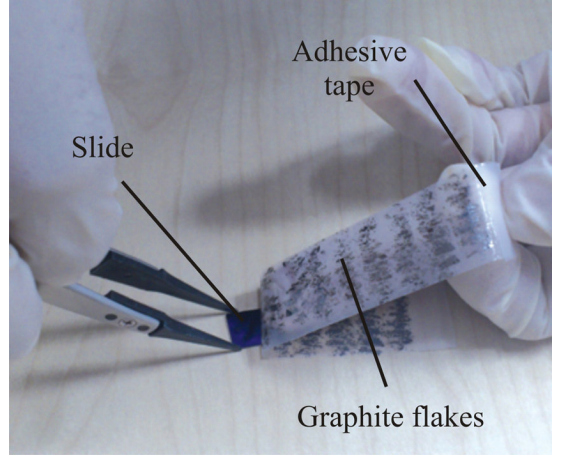


FIG. 3: Scotch Tape exfoliation method example [23].

2. Data Processing

In this step AFM data is analyzed using Gwyddion and flakes are labeled in a primitive way using paint, classifying all the different colors and regions of flakes. Only flakes that have been measured with the AFM are labeled. Regions of flakes that are thicker than $28nm$ are ignored. In figure 4 (c) an example is shown.

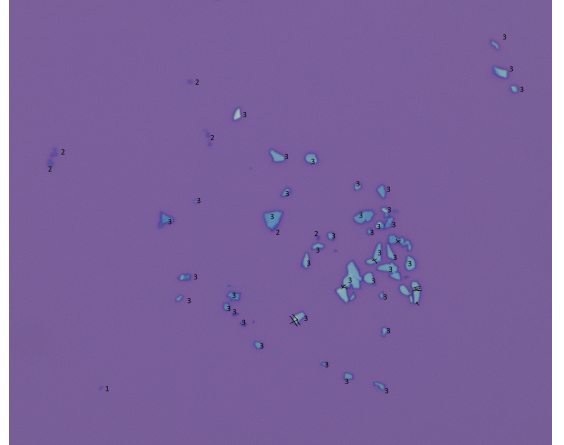


FIG. 4: Data processing stage example in Paint.

3. Model Assisted Labeling (MAL)

After processing the data, MRCNN is used to make predictions. These predictions are saved as pre-labels and uploaded to Labelbox using the Labelbox API. In figure 5 (d) the MAL predictions are shown for the example image. The pre-labels will be corrected and adjusted in the next step. This results in less flakes to be manually labeled. The used weights for MRCNN in this step may be the weights of one of the starting materials or the

weights of an intermediate training run.

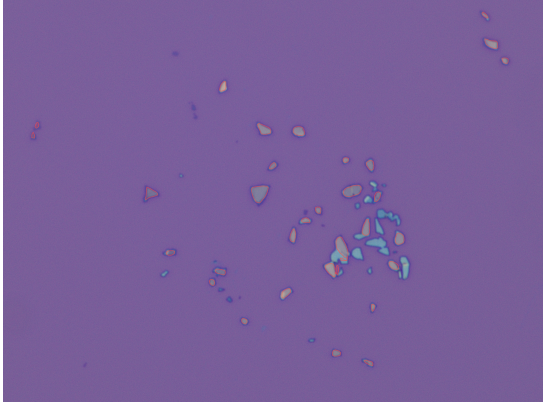


FIG. 5: Model assisted labeling stage example in Labelbox.

4. Labeling

Labeling consists of classifying and creating a segmentation mask for each primitively labeled flake, as well as correcting pre-labels when necessary. A segmentation mask is created by manually selecting the coordinates that form the boundary of a mask. Figure 6 (e) shows the final labeling result. A flake that contains different thickness regions, that fall in the same bin, are labeled as one region. All flakes with AFM data are labeled, regardless of their size.

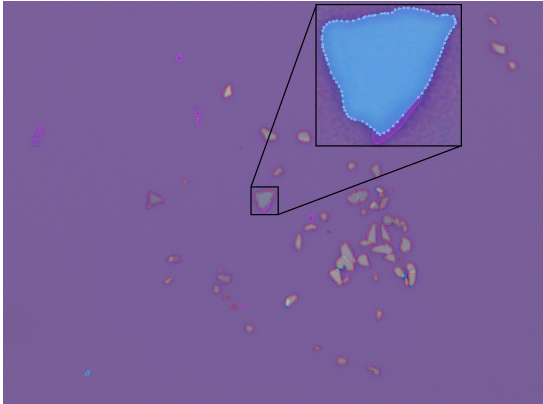


FIG. 6: Data labeling stage example in Labelbox.

5. Image Splitting

Image splitting splits images with many objects into multiple images (called split images) with few objects. This is done by extracting flakes from images and pasting them on backgrounds (images without flakes). Only

flakes that have been labeled can be extracted. It is important to extract flakes in the correct context, such that the split images are realistic. For example, two flakes too close to each other must be extracted together. Otherwise each new image could contain only half a flake.

Only images with markers and an annotation amount above a certain threshold are split. To check whether an image contains a marker, the HSV color-space is analyzed. The algorithm loops through all the annotations of an image. If an annotation is already part of another split image it is skipped. Then a search area is created, which is the bounding box extended by a certain amount of pixels. This area is used to check if there is any overlap with other annotations, to preserve the correct context. Found overlapping annotations are added to the split image, and a new search area is created to check for new overlapping annotations. This search area overlap check is iterated until no overlapping annotations are found. Sometimes a new split image completely overlaps an older split image. If this happens, the annotations from the older image are moved to the new image. This results in zero annotations for the older image. The annotations of the new split image are saved. And the split image is created by cutting out the right part of the original image and pasting it on a random background. After analyzing all the annotations of an image, the split images with zero annotations are deleted. An example of the algorithm is shown in figure 7.

Image splitting has two reasons. The first one being computational bottlenecks. As earlier mentioned, MR-CNN makes predictions for each identified RoI, resulting in high computational costs for images with many RoIs. Meaning images with many flakes. This can result in long computations and high memory usage. The second reason is the generalization of the model. By extracting flakes to random backgrounds, the model sees flakes in more different contexts. Also resulting in a significant increase of the dataset. With very limited AFM data, but in some images many measured flakes, overfitting can be suppressed.

For the split images to be realistic, the background color from the original image (of which the flakes are extracted) and the new background must match. The images are taken with auto exposure, this causes all images at zoom 100x with a yellow marker to have the same background color. For images that did not contain a marker it was not possible to get matching backgrounds. That is why the algorithm checks if the image contains a marker for image splitting. This results in around 40 less images after image splitting without the unmarked images.

6. Data Split

The dataset is split into three parts: (i) train, 80% - 199 images, used for training the model, (ii) validation, 10% - 25 images, used to tune the hyperparameters

during training and (iii) test, 10% - 25 images, for an unbiased evaluation after training. In table I the amount of objects in each class for each data split are shown.

	Class 1	%	Class 2	%	Class 3	%
Train	4	1	163	22	559	77
Validation	1	1	23	26	63	72
Test	0	0	17	27	46	73

TABLE I: Amount of objects in each class for all three data splits.

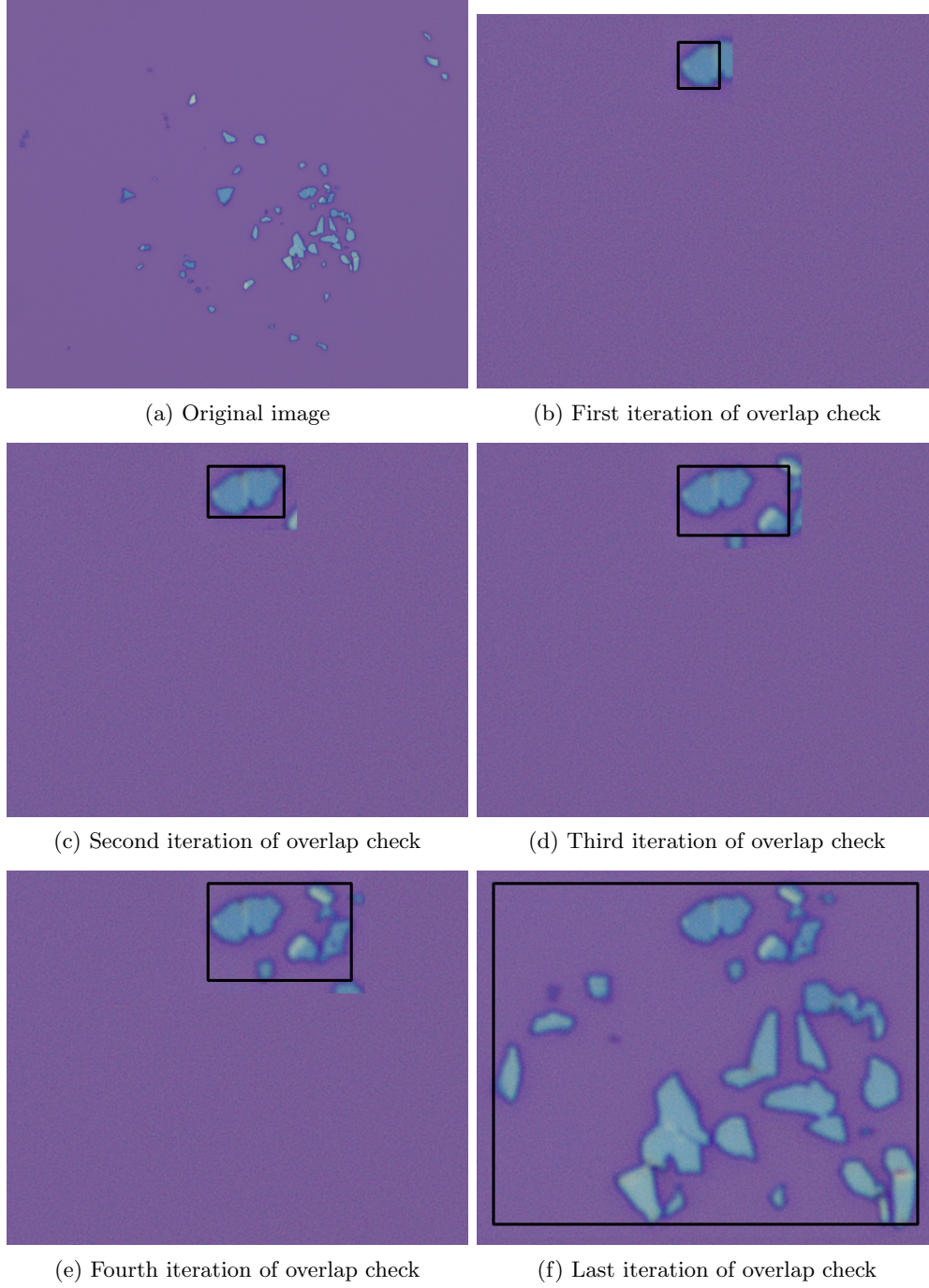


FIG. 7: Image splitting algorithm overlap iterations.

B. Training

The control parameters of equation 1 are set to $(\alpha, \beta, \gamma) = (0.6, 1.0, 1.0)$.

Training, from each starting material, took around 7 hours, using 1 GPU (NVIDIA Tesla A100 GPU with 80-GB memory). The training consists of four stages. Each stage takes 30 epochs. An epoch is a training iteration where the model trains on all the images in the train dataset.

- (i) Train network heads. $lr = 10^{-3}$, the last layers of MRCNN that make the predictions.
- (ii) Fine-tune the last part of the backbone and up. $lr = 10^{-4}$.
- (iii) Fine-tune all layers. $lr = 10^{-4}$.
- (iv) Fine-tune all layers. $lr = 10^{-5}$.

Training starts with only the last part of the model, and during training the amount of layers is increased and the learning rate is decreased. Two other important hyperparameters are the batch size and momentum, set to 4 and 0.9 respectively. During training the weights of MR-CNN are updated according to the calculated loss. And this loss is calculated for each batch, the average loss of four images. The batch size determines the impact every image has on the weights.

The batch size and momentum both influence the path the model takes through the loss landscape, looking for the absolute minimum. A too large batch size mitigates stochastic updates of the weights, and thus can prevent jumping out of local minima.

Now imagine the model to be a ball rolling through the loss landscape, the momentum parameter increases the mass of this ball. This is done by considering the previous trajectory the model took through the landscape, and using that to influence the direction it is heading in.

During training the model repeatedly saves its new best weights. These are the weights where the validation loss has reached a new minimum. This is a form of early stopping. A machine learning regularization technique used to prevent overfitting, done saving the weights before the overfitting starts.

C. Evaluation

There are many ways to evaluate the performance of machine learning models. During training, for each epoch, the loss on the training and validation dataset are saved. These plots are called training curves. They are mainly used to analyze whether the model is over- or underfitting, or neither. Training curves are not a direct measure of the models performance, but a qualitative indication that it might be improving or not.

After training the test dataset is used for an unbiased evaluation, done by using the COCO evaluation. An industry standard evaluation for computer vision models. And used in many research papers [24]. This evaluation calculates the average precision (AP) and average recall

(AR) for different intersection over union (IoU) thresholds and mask sizes. Note that COCO does not differentiate between the mean average precision (mAP) and the average precision (AP). The precision and recall are defined as follows:

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

- True Positives (TP): Predicted a label and matches the ground truth.
- False Positives (FP): Predicted a label and does not match the ground truth.
- False Negatives (FN): Did not predict a label, while there was a label in the ground truth.

The precision measures how many predictions are correct. The recall measures how many of the labels in the ground truth are predicted correctly.

The IoU is defined as:

$$IoU = \frac{Area\ of\ overlap}{Area\ of\ union} \quad (8)$$

A schematic explanation is shown below in figure 8.

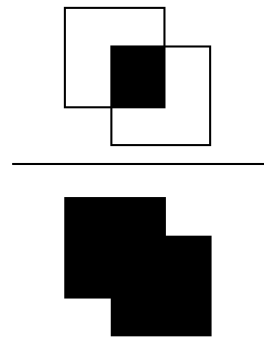


FIG. 8: IoU calculation visualization.

COCO considers three different IoU evaluations: (i) ≥ 0.50 (PASCAL VOC metric), (ii) ≥ 0.75 (strict metric) and (iii) $0.50 : 0.95$ (primary challenge metric), an average of different IoU thresholds ranging from 0.50 to 0.95 with step size 0.05. Thus a prediction must have an IoU greater than 0.50 to be considered a detection. The third evaluation is the most robust.

To evaluate the models performance for each class, a confusion matrix is created. This is done manually on the test dataset. Comparing predictions to ground truths for each class. This method does not consider different mask sizes and IoU thresholds.

IV. RESULTS AND DISCUSSION

A. Starting Material Comparison

The starting materials are compared by running the AP@IoU_{50%:95%} and AR@IoU_{50%:95%} COCO evaluation on the test NbSe2 dataset. This is done for the trained and untrained weights. The COCO evaluation computes these scores for different flake sizes. The sizes are defined in px^2 , the number of pixels in the segmentation mask. Small: $0 - 32^2 px^2$, medium: $32^2 - 96^2 px^2$ and large: $> 96^2 px^2$. There are multiple definitions for object sizes in the machine learning industry, but these are the ones that COCO uses. At zoom 100x the resolution is $0.048 \mu m/px$. Converting this to our size bins: $32px \leftrightarrow 1.54 \mu m$, $96px \leftrightarrow 4.61 \mu m$. The comparison of sizes is done in px^2 because this is independent of the microscope resolution. Otherwise, a model may seem to outperform other models on flakes smaller than $1.54^2 \mu m^2$, while it simply has a higher resolution microscope.

In figure 9 the results of the starting material weights are shown, meaning the untrained weights.

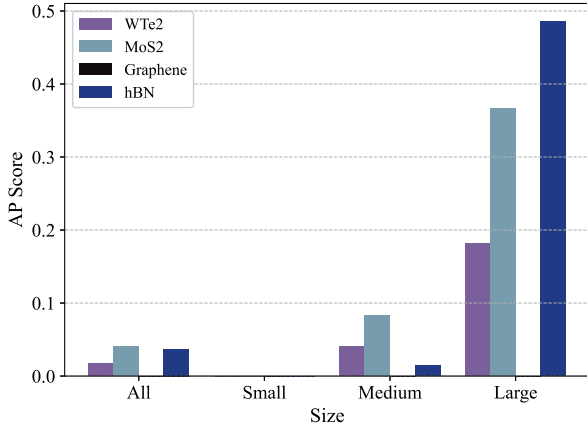


FIG. 9: COCO Evaluation of the starting weights. AP score for IoU 0.50:0.95 and different area sizes for each starting material.

Graphene did not get any predictions correct and there are no correct predictions for small flakes. Looking at all sizes, MoS2 and hBN perform about the same. And there is a huge performance difference between the different flake sizes. Considering the material properties discussed in the introduction, hBN should not perform this well compared to WTe2. This shows that there might be some similarities between the starting material datasets and the NbSe2 dataset, that apparently have a larger impact on the models results than the band gaps of these materials. Possible similarities could be flake sizes, lighting conditions, or background artifacts.

From this graph one could expect hBN or MoS2 to

result in best trained weights. In figure 10 the COCO evaluation results of the trained weights are shown.

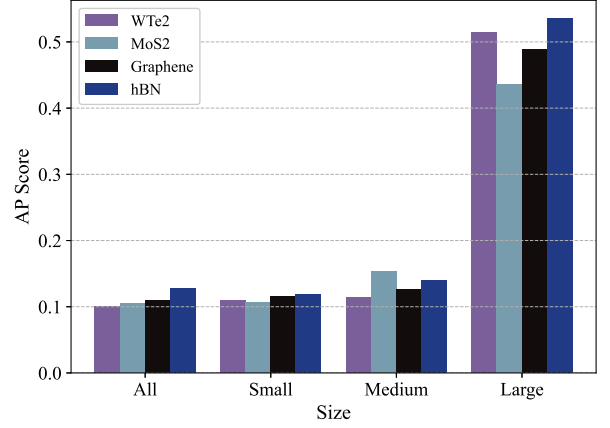


FIG. 10: COCO Evaluation of the trained weights. AP score for IoU 0.50:0.95 and different area sizes for each starting material.

All materials perform approximately the same, with hBN being slightly better. Again there is a huge difference in the performance between the large flakes and small and medium flakes. Graphene made the biggest improvement compared to the untrained results. And the small and medium flakes have about the same performance results.

In figure 11 the AR scores of the untrained starting materials are shown.

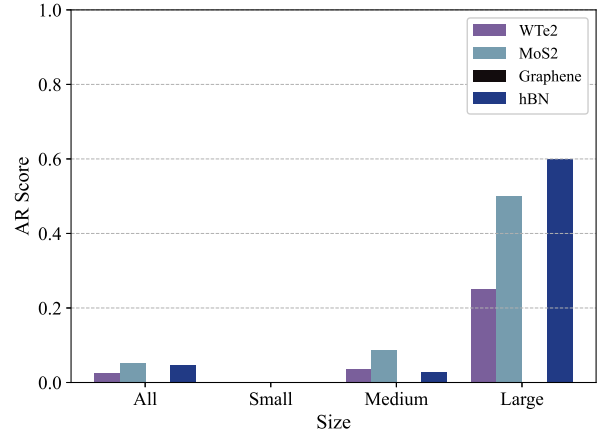


FIG. 11: COCO Evaluation of the starting weights. AR score for IoU 0.50:0.95 and different area sizes for each starting material.

There are no ground truth labels predicted correctly by the graphene starting weights and for small flakes. For all sizes MoS2 and hBN perform about the same.

And there is a big performance difference between the different flake sizes. Again it is interesting to see hBN performing this well compared to the other materials. In figure 12 the AR scores of the trained starting materials are shown.

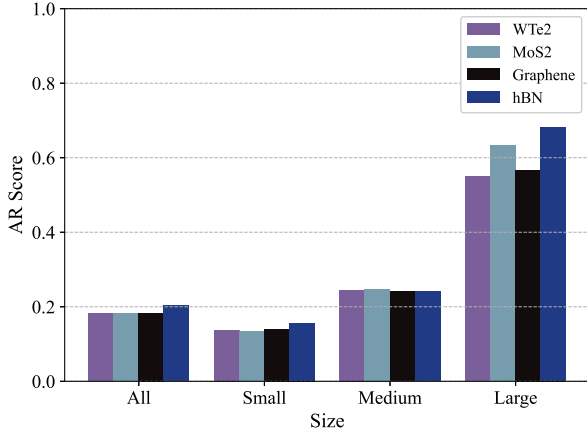


FIG. 12: COCO Evaluation of the trained weights. AR score for IoU 0.50:0.95 and different area sizes for each starting material.

After training all the material perform about the same, with again hBN having a slight advantage. The performance of graphene has improved. Predictions on small flakes also improved, but medium flakes still score higher. And again noting the difference between the large flakes and small and medium flakes.

Considering the trained AP and AR scores, the starting material does not matter after training. Which is really interesting, because the performance of transfer learning should be strongly dependent on the similarity between datasets. Figures 9 and 11 clearly show that there is a difference between the starting material datasets and the NbSe2 dataset.

From now on, unless mentioned otherwise, the hBN starting weights are used.

1. Training Curves

Training curves can be used for a qualitative analysis of the training. For example to check whether the model is over- or underfitting. Overfitting is expressed as a strong divergence between the train and validation curves. Underfitting can be spotted if there is a large gap between the curves while they are still converging.

It is important to mention that the loss is not a direct measure of performance. In figures 13, 14 and 15 the losses of the bbox, class and mask are shown respectively.

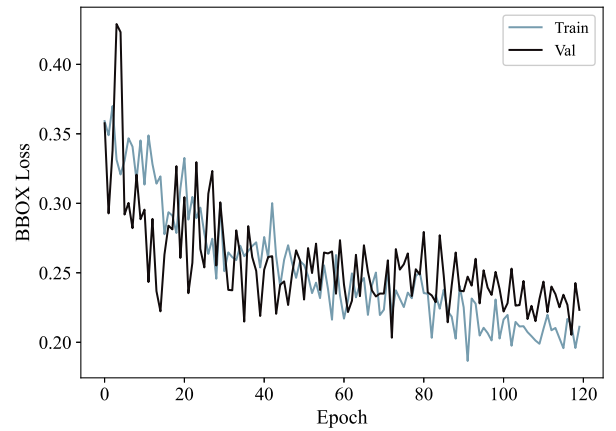


FIG. 13: Training curve of the bbox loss, starting from hBN.

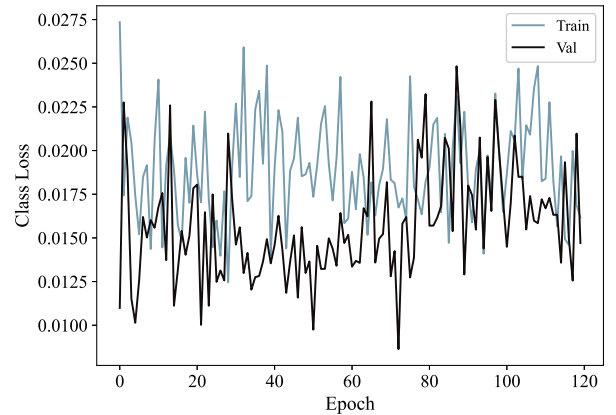


FIG. 14: Training curve of the class loss, starting from hBN.

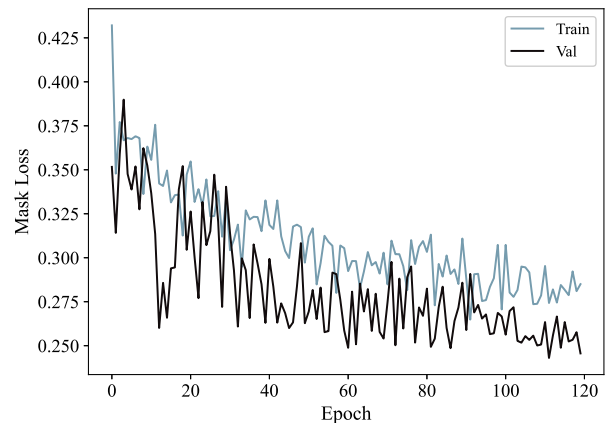


FIG. 15: Training curve of the mask loss, starting from hBN.

The train and validation curves in the plots above overlap quite a lot and there seems to be only a slight sign of

overfitting after around 110 epochs in the mask loss curve (figure 15). And no signs of underfitting. It is interesting that the class loss does not decrease, which could be a result of the very unbalanced dataset.

B. Predictions

In the appendix (section VIC) three test images are shown with model predictions and ground truth values.

C. Flake Sizes

One of the important results of the COCO evaluation is the big difference in performance for different flake sizes, discussed in section IV A. To investigate this further a small modification is made to the COCO evaluation. This is done to compare evaluation results to other research papers. The modification consists of adding the AP and AR evaluation for different flake sizes for an IoU threshold of 0.50 (AP@IoU_{50%} and AR@IoU_{50%}). Results are shown in table II.

px^2	$0 - 32^2$	$32^2 - 96^2$	$> 96^2$	All
AP	0.225	0.221	0.850	0.231
AR	0.241	0.318	1.000	0.239

TABLE II: COCO evaluation AP and AR scores for an IoU threshold of 0.50 for different sizes. Trained from hBN.

[9] reached an AP@IoU_{50%} (for all flake sizes) of 0.49 for graphene and 0.52 for WTe2. Stating “..these values are sufficiently high and can be successfully applied to searches for 2D crystals.” Comparing this to table II, the new NbSe2 model can be successfully be applied for large 2D crystal searches. It is important to mention that the dataset split of [9] only consisted of a train and validation set, not a test set. Their final COCO evaluation results are from the validation set. But since the validation set is also used during training, their results are not unbiased. And thus an actual unbiased test evaluation could be lower than their reported values.

[25] trained a machine learning model for Graphene and WSe2. And evaluated the model on images with zoom 20x (resolution $0.25\mu m^2/px^2$). Their COCO evaluation results for single layer flakes are shown in table III.

px^2	$14^2 - 20^2$	$20^2 - 28^2$	$28^2 - 40^2$	$> 40^2$
AP Graphene	0.319	0.854	0.887	0.964
AP WSe2	0.527	0.785	0.778	0.1000
AR Graphene	0.827	0.933	0.933	0.1000
AR WSe2	0.623	0.819	0.810	0.1000

TABLE III: COCO evaluation AP and AR for IoU threshold of 0.50 for different sizes of [25].

Comparing table III to table II, [25]s models performs significantly better for small and medium objects and slightly better for large objects. Again concluding that the trained NbSe2 model work well for large flakes, but not for small and medium flakes.

Poor performance for small objects is a common issue for MRCNN. A possible cause is that the edge to area ratio in images of small flakes is relatively large [25]. Different data augmentation strategies and new backbone networks have proposed to improve small object detection [26] [27] [28]. In figure 16 the distribution of flake sizes of the entire dataset is plotted.

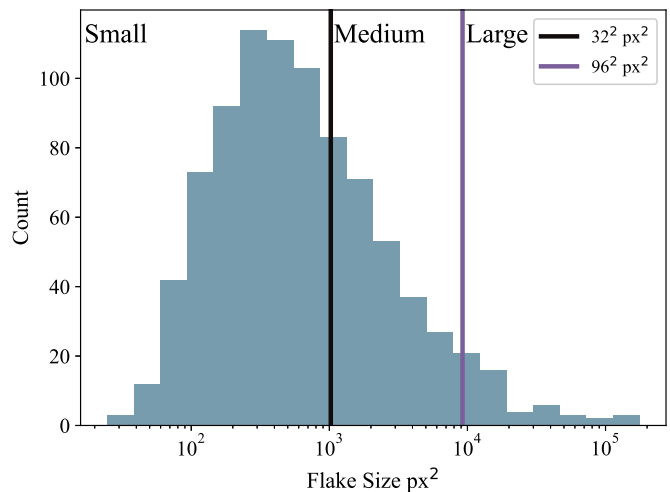


FIG. 16: Flake size distribution of entire dataset.

This shows that the small and medium flakes are not underrepresented, but the large flakes are. Which is also interesting since the model performs much better for the large flakes. This could be caused by the flake size distribution of the ‘parent’ dataset in the transfer learning process. The 2D materials dataset of [9] could for example mainly consist of large flakes.

This difference in model performance for each flake size does not necessarily impact the usage of deep learning for flake searching and classification. It completely depends on the research. For example if flakes smaller than $4.61\mu m$ by $4.61\mu m$ are discarded for research, it does not matter that the model does not perform well on these flakes.

To increase the models performance on flakes smaller than $4.61\mu m^2$ a higher resolution camera can be used.

D. Class Performances

To investigate how the unbalanced dataset influences the model a confusion matrix is created. This evaluation does not consider IoU thresholds or flake sizes, and is done manually. The horizontal axis represents the models predictions, and the vertical axis represents the actual ground truth value. The result is shown in table IV.

	Class 1	Class 2	Class 3	BG
Class 1	0	0	0	0
Class 2	0	1	9	9
Class 3	0	0	34	12
BG	0	9	46	NA

TABLE IV: Confusion matrix of the test dataset. Horizontal axis represents predictions and the vertical axis the ground truth. BG = background. Trained from hBN.

The resulting precision and recall, calculated from this confusion matrix, are shown in table V.

	Precision	Recall
Class 1	0	0
Class 2	0.1	0.1
Class 3	0.382	0.739

TABLE V: Precision and recall per class, calculated from the confusion matrix.

The model did predict some class 2 flakes as class 3, but never predicted a class 3 flake as class 2. And the model frequently predicts regions in the image as either class 2 or 3 (much more often class 3), when they should be ignored, meaning predicted as BG. This clearly shows that the model is biased towards Class 3. Which can also be seen from the precision and recall values in table V.

The best way to solve this is to create a dataset where every class is represented equally. But data of class 1 flakes can be difficult to obtain. One could also split a dataset into a balanced and unbalanced set. Then use the unbalanced dataset as a transfer learning step. And then fine-tuning that resulting model on the balanced dataset. It is also possible to use data augmentation techniques specifically on the underrepresented class.

E. Image Splitting

1. Training

All the earlier mentioned results make use of image splitting to increase the dataset. Therefore it is interesting to investigate what happens without image splitting. Table VI shows the influence of image splitting on training. It compares an early training run (comparison 1)

on 36 images, before the dataset was complete, to final a training run (comparison 2) on 80 images. Both starting from MoS2. It is important to mention that this entire image splitting evaluation is very qualitative.

	Splitting	Images	Time	Max Memory
Comparison 1	Without	36	2h54m	81Gb
	With	92	3h7m	21Gb
Comparison 2	Without	80	6h32m	38Gb
	With	249	7h7m	38Gb

TABLE VI: With and without image splitting training comparison. Images, training time and max memory usage.

For comparison 1 there is a huge difference in max memory usage. Which could mean that the computational bottleneck from having to many RoIs in an image is mitigated. But since the max memory usage is the same for the final run, with and without image splitting, the presence of a computational bottleneck during training probably is dependent on whether a specific image is either in the train and validation set or the test set.

2. Training Curves

The training curves can be analyzed to see if the image splitting helps mitigating overfitting. In figures 17, 18 and 19 the losses of the bbox, class and mask are shown respectively for comparison 1.

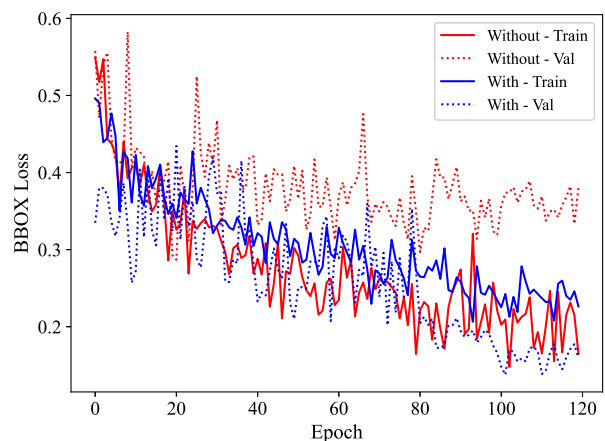


FIG. 17: Comparison 1 training curves of the bbox loss, with and without image splitting. Starting from MoS2.

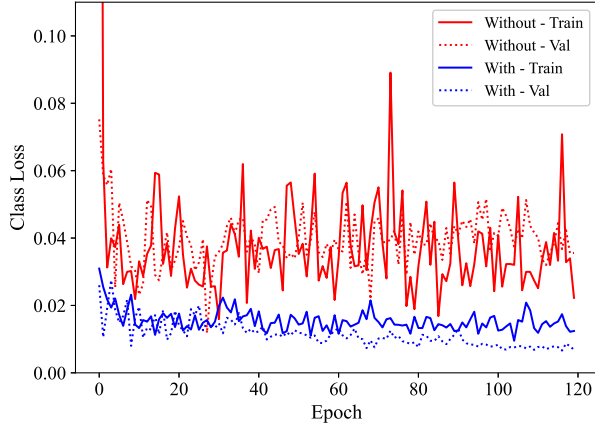


FIG. 18: Comparison 1 training curves of the class loss, with and without image splitting. Starting from MoS2.

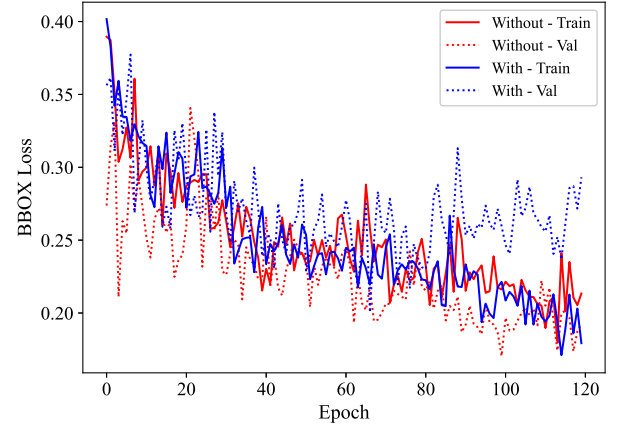


FIG. 20: Comparison 2 training curves of the bbox loss, with and without image splitting. Starting from MoS2.

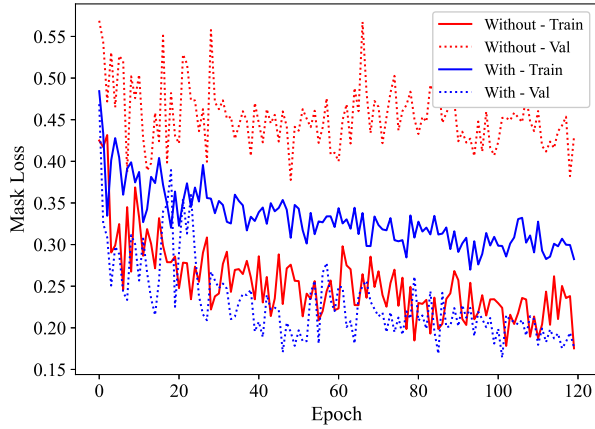


FIG. 19: Comparison 1 training curves of the mask loss, with and without image splitting. Starting from MoS2.

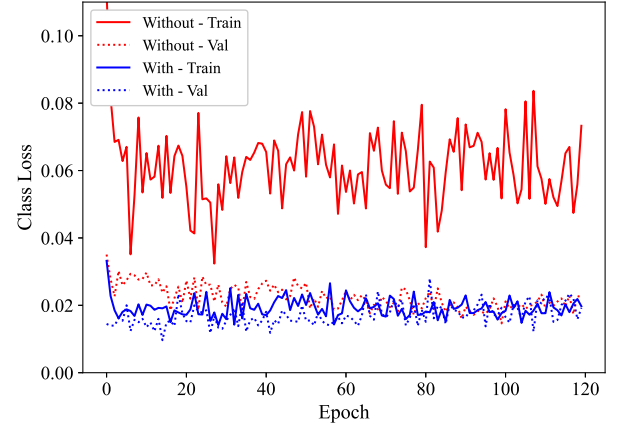


FIG. 21: Comparison 2 training curves of the class loss, with and without image splitting. Starting from MoS2.

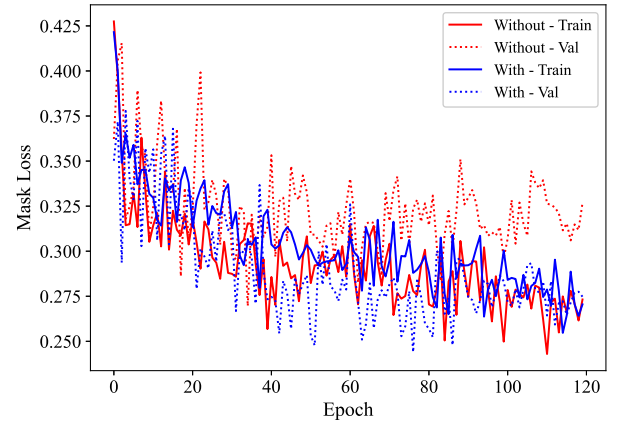


FIG. 22: Comparison 2 training curves of the mask loss, with and without image splitting. Starting from MoS2.

The class loss stays pretty constant, neither run overfits and the train and validation losses are close together. For the bbox and mask loss the runs with image splitting are much closer together and diverge less than the runs without image splitting. This shows that the image splitting in this case helps preventing overfitting.

In figures 20, 21 and 22 the losses of the bbox, class and mask are shown respectively for comparison 2.

For this comparison the results are different. With

the mask loss, image splitting prevents overfitting. But for the bbox los image splitting enables overfitting. The class loss without image splitting might show underfitting, since the train and validation loss are not close together and neither diverge or converge. It seems as if the step from 36 to 92 images has a great positive impact, and the step from 80 to 249 images does not really matter, considering the training curves. This might be caused by a lack of diversity in the dataset. Adding images that look alike will not contribute to the learning process [29].

3. COCO Evaluation

The evaluation results are only available for comparison 2. In figure 23 the AP COCO evaluation results are shown for comparison 2. An extra result is added, a model trained starting from hBN.

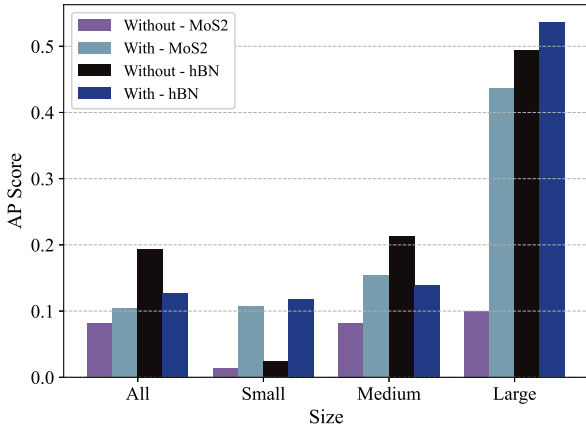


FIG. 23: AP COCO evaluation results of comparison 2. Trained from MoS2 and hBN.

Something very interesting is happening in this plot. The starting material comparison shows that after training it does not matter which starting material is used. Which is a strange result because of how transfer learning works. In the figure above the AP values of MoS2 significantly increase with the use of image splitting. That is not the case for hBN, for medium sized flakes the performance even drops. Notice that hBN performs much better without image splitting than MoS2. It looks like image splitting bridges the gap of different material properties in transfer learning. This could be due to the overall large increase in images. That after the image increase the model is able to converge to a loss in the loss landscape, regardless of the starting material.

In figure 24 the AR COCO evaluation results are shown.

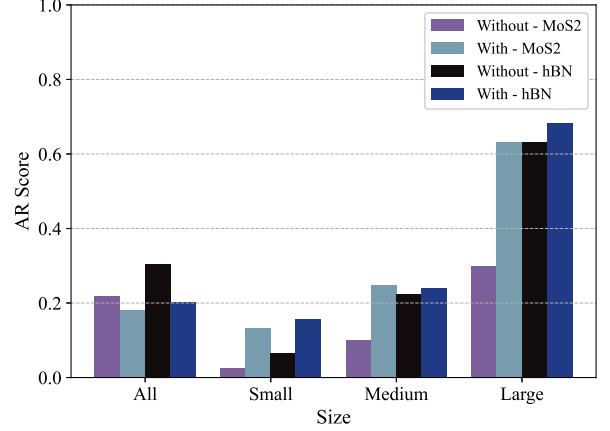


FIG. 24: AR COCO evaluation results of comparison 2. Trained from MoS2 and hBN.

Looking at the small, medium and large sizes, the same analysis holds as for the AP scores. The scores for the all sizes category in this plot do not add up. Both the without scores are higher than the with scores, while for each individual category this is not the case.

F. Extended Dataset

Adding an extra class could be valuable for research where very thick flakes are required. And often the model classifies a flake that is too thick, $> 28nm$, as class 3, instead of background. To investigate this further a second dataset is created with four thickness bins. Adding class 4, with thickness $> 28nm$. This is done with the idea that $p(class\ 4) > p(class\ 3) > p(BG)$. For too thick flakes for the original dataset. Adding an extra class, while in the first case only having 3 classes, significantly increases the complexity of the problem. And in consequence requiring more data. In figures 25 and 26 the COCO evaluation scores are shown for a model trained on this extended dataset, starting from hBN. They are compared to the performance of the not extended dataset.

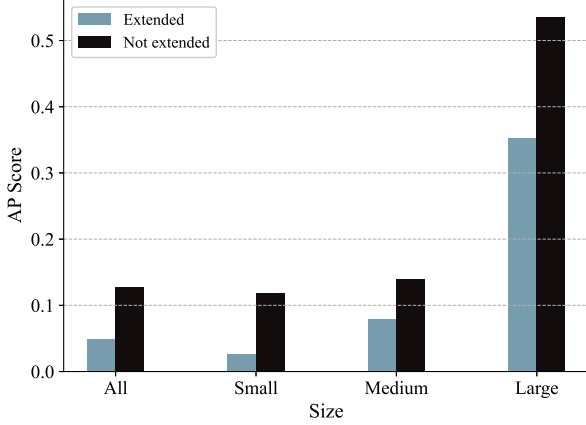


FIG. 25: COCO Evaluation of the extended dataset versus the not extended dataset. AP scores for IoU 0.50:0.95 and different area sizes. Starting from hBN

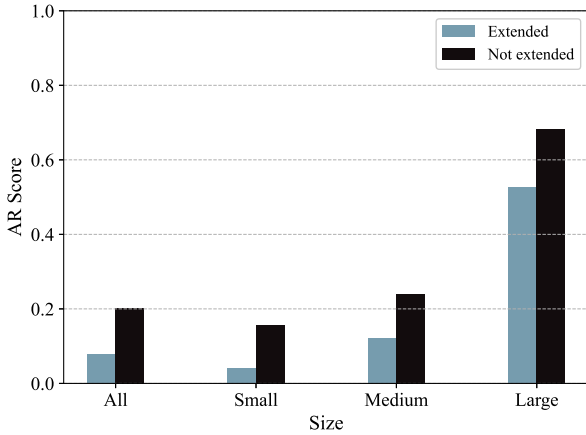


FIG. 26: COCO Evaluation of the extended dataset versus the not extended dataset. AR scores for IoU 0.50:0.95 and different area sizes. Starting from hBN

As expected the extended dataset performs worse in all size categories than the not extended dataset. Apart from the dataset probably not being large enough for four classes, there is another cause for diminished performance. The last layer of the neural network that outputs the class prediction is directly linked to the amount of classes. In the case of classes class 1, class 2, class 3 and BG it contains four neurons. Adding an extra class to the dataset, adds an extra neuron to the network. This changes the dimensions of the layer. Therefore the pre-

trained weights from [9] cannot be used for this layer. This results in even more data being needed.

V. CONCLUSION

The use of image splitting diminishes the influence of different starting materials on the performance of the final model. hBN results in the best NbSe2 model, which is unexpected considering the optical properties of both materials. The training curves show almost no overfitting. Large flakes score sufficiently high ($\text{AP@IoU}_{50\%:95\%}$ of 0.54) to be used for automated flake searching. The small ($\text{AR@IoU}_{50\%:95\%}$ of 0.12) and medium ($\text{AR@IoU}_{50\%:95\%}$ of 0.14) flakes score significantly lower. Because of the unbalanced dataset, the final model is very biased towards class 3. Adding an extra class does not improve the model due to a lack of data.

There are multiple things that should be investigated further. (i) The starting material hBN leads to the best NbSe2 model. Clearly something more is going on than the comparison of optical properties between materials. (ii) A balanced dataset increases the precision and recall for each class.

The validity of image splitting could also be questioned since the created images do not perfectly match reality. Additionally, the test dataset contains image splitting images. This means that a model trained with image splitting is not yet tested on only unsplit images. The results are likely to be almost identical.

The data acquisition stage of this project is very time consuming. Automating AFM measurements could significantly increase the dataset. With a larger dataset adding extra classes could be further explored. By performing AFM measurements with a high enough resolution, one could overlap the AFM image with the optical image of a flake. And from that overlap extract the mask coordinates. This will speed up the data acquisition process even more, because manually labeling flakes is also very time consuming.

For training, the hyperparameters could be further fine-tuned to improve performance. Hyperparameters have a large influence on training and the final results, but this was not a point of interest for this project.

This project allows researchers, after creating a balanced dataset, to build an automated flake searching setup for NbSe2. The created code infrastructure for data processing, training and evaluation makes it possible to train models for new materials with minimal additional effort.

-
- [1] Y. Cao, V. Fatemi, and S. Fang et al. Elements beyond graphene: Current state and perspectives of elemental monolayer deposition by bottom-up approach. <https://doi.org/10.1016/j.apmt.2019.100502>, 2020.
 - [2] V. Shanmugam, R. Afriyie Mensah, and et al. A review of the synthesis, properties, and applications of 2d materials. <https://doi.org/10.1002/ppsc.202200031>, 2022.
 - [3] Santosh K. Tiwari, Sumanta Sahoo, and et al. Graphene research and their outputs: Status and prospect. <https://doi.org/10.1016/j.jsamd.2020.01.006>, 2020.
 - [4] A. Geim and I. Grigorieva. Van der waals heterostructures. <https://doi.org/10.1038/nature12385>, 2013.
 - [5] N. Antonatos, H. Ghodrat, and Z. Sofer. Unconventional superconductivity in magic-angle graphene superlattices. <https://doi.org/10.1038/nature26160>, 2018.
 - [6] P. Kumbhakar, J.S. Jayan, and A.S. Madhavikutty et al. Prospective applications of two-dimensional materials beyond laboratory frontiers: A review. <https://doi.org/10.1016/j.isci.2023.106671>, 2023.
 - [7] A. Gupta, A. Anpalagan, and L. Guan et al. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. <https://doi.org/10.1016/j.array.2021.100057>, 2021.
 - [8] T. Falk, D. Mai, and R. Bensch et al. U-net: deep learning for cell counting, detection, and morphometry. <https://doi.org/10.1038/s41592-018-0261-2>, 2019.
 - [9] S. Masubuchi, E. Watanabe, and Y. Seo et al. Deep-learning-based image segmentation integrated with optical microscopy for automatically searching for two-dimensional materials. <https://doi.org/10.1038/s41699-020-0137-z>, 2020.
 - [10] TY. Lin and et al. Microsoft COCO: Common objects in context. https://doi.org/10.1007/978-3-319-10602-1_48, 2014.
 - [11] M. Bernico, Dingchao Zhang Y. Li, and et al. Investigating the impact of data volume and domain similarity on transfer learning applications. <https://arxiv.org/abs/1712.04008>, 2018.
 - [12] K. Dwivedi, J. Huang, and et al. Duality diagram similarity: a generic framework for initialization selection in task transfer learning. <https://doi.org/10.48550/arXiv.2008.02107>, 2020.
 - [13] T. Zhu, P.M. Litwin, and Md.G. Rosul et al. Transport properties of few-layer nbse2: From electronic structure to thermoelectric properties. <https://doi.org/10.1016/j.mtphys.2022.100789>, 2022.
 - [14] Y. Duan, C.D. Stinespring, and B. Chorpene et al. Electronic structures, bonding configurations, and band-gap-opening properties of graphene binding with low-concentration fluorine. <https://doi.org/10.1002/open.201500074>, 2015.
 - [15] W. Wen, Y. Song, and X. Yan et al. Recent advances in emerging 2d nanomaterials for biosensing and bioimaging applications. <https://doi.org/10.1016/j.mattod.2017.09.001>, 2018.
 - [16] S. Castelletto, F. A. Inam, and S. i. Sato et al. Hexagonal boron nitride: a review of the emerging material platform for single-photon sources and the spin-photon interface. <https://doi.org/10.3762/bjnano.11.61>, 2020.
 - [17] C. Elias, P. Valvin, and T. Pelini et al. Direct band-gap crossover in epitaxial monolayer boron nitride. <https://doi.org/10.1038/s41467-019-10610-5>, 2019.
 - [18] S. Puebla, H. Li, and H. Zhang et al. Apparent colors of 2d materials. <https://doi.org/10.1002/adpr.202100221>, 2022.
 - [19] Y. Mingxiao, W. Dustin, and Z. Dongyan et al. Recent advancement on the optical properties of two-dimensional molybdenum disulfide (mos2) thin films. <https://doi.org/10.3390/photonics2010288>, 2015.
 - [20] CH. Lee, E. Silva, and L. Calderin et al. Tungsten ditelluride: a layered semimetal. <https://doi.org/10.1038/srep10013>, 2015.
 - [21] K. He, G. Gkioxari, and P. Dollár et al. Mask R-CNN. <https://ieeexplore.ieee.org/document/8237584>, 2017.
 - [22] R. Girshick. Fast R-CNN. <https://ieeexplore.ieee.org/document/7410526>, 2015.
 - [23] E. Randviir, D. Brownson, and et al. A decade of graphene research: Production, applications and outlook. <https://doi.org/10.1016/j.mattod.2014.06.001>, 2014.
 - [24] G. Alfonso-Francia, J.C. Pedraza-Ortega, and et al. Performance evaluation of different object detection models for the segmentation of optical cups and discs. <https://doi.org/10.3390/diagnostics12123031>, 2022.
 - [25] J. Uslu, T. Ouaj, and et al. An open-source robust machine learning platform for real-time detection and classification of 2d material flakes. <https://doi.org/10.1088/2632-2153/ad2287>, 2024.
 - [26] M. Kisantal, Z. Wonja, and et al. Augmentation for small object detection. <https://doi.org/10.48550/arXiv.1902.07296>, 2019.
 - [27] J. Peng, Y. Zhao, and et al. Dsaff-net: A backbone network based on mask r-cnn for small object detection. <https://doi.org/10.32604/cmc.2023.027627>, 2022.
 - [28] C. Jiang, T. Zhang, and et al. Research on small target detection algorithm of uav based on mask r-cnn. <https://doi.org/10.1145/3603781.3604216>, 2023.
 - [29] M. Toneva, A. Sordoni, and et al. An empirical study of example forgetting during deep neural network learning. <https://doi.org/10.48550/arXiv.1812.05159>, 2019.

VI. APPENDIX

A. Datasets

Material	Images	Objects	Total	Objects Class 1	Objects Class 2	Objects Class 3
hBN	232	274	0	0	274	
Graphene	862	4805	1858	2081	866	
MoS2	569	839	239	523	77	
WTe2	318	1053	148	582	323	
Total	2102	7153	2258	3297	1598	

TABLE VII: Amount of images and object for each material in the original dataset of [9].

B. Code

The code is available at https://github.com/Linuxable/tdmms_DL.

C. Model Predictions

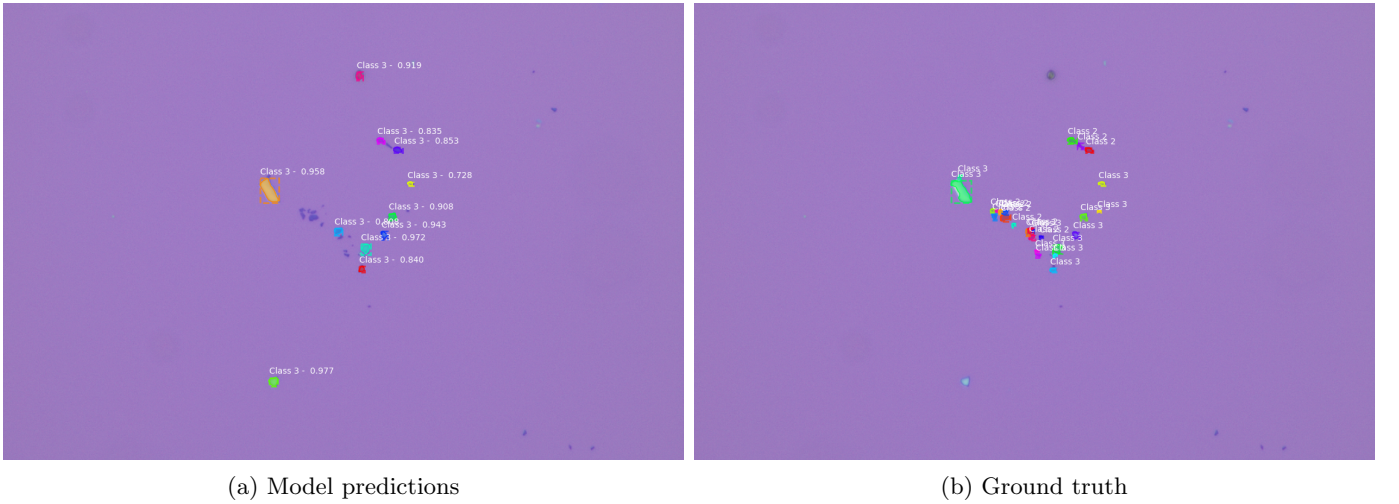


FIG. 27: Model predictions (a) and ground truth (b) of a test image. The value next to the class name in the predictions is the confidence value of the model.

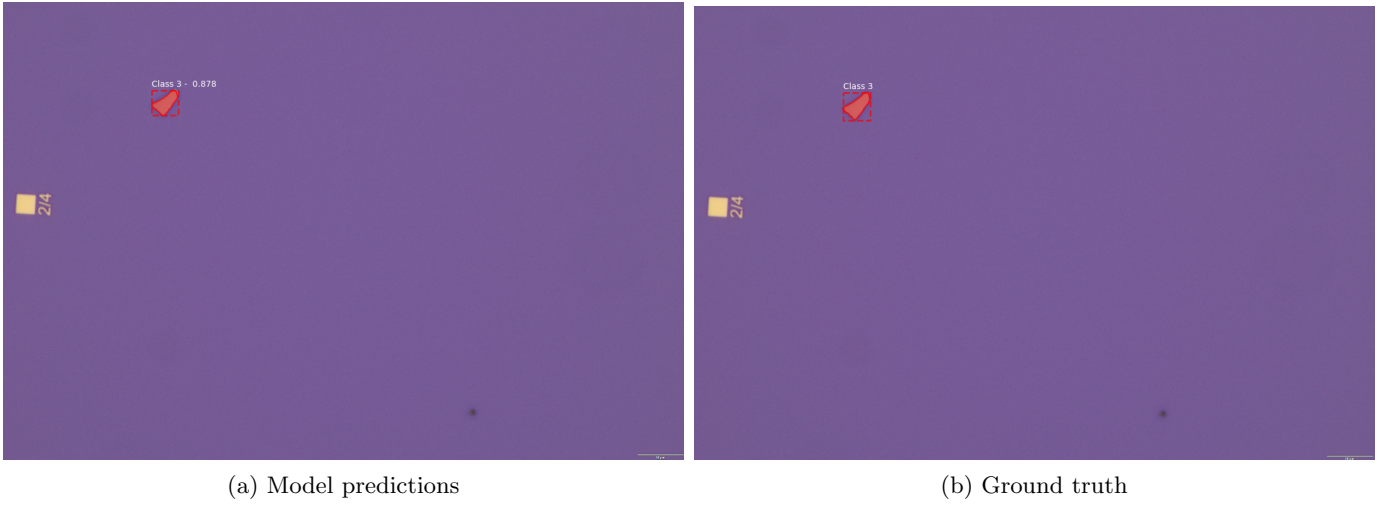


FIG. 28: Model predictions (a) and ground truth (b) of a test image. The value next to the class name in the predictions is the confidence value of the model.

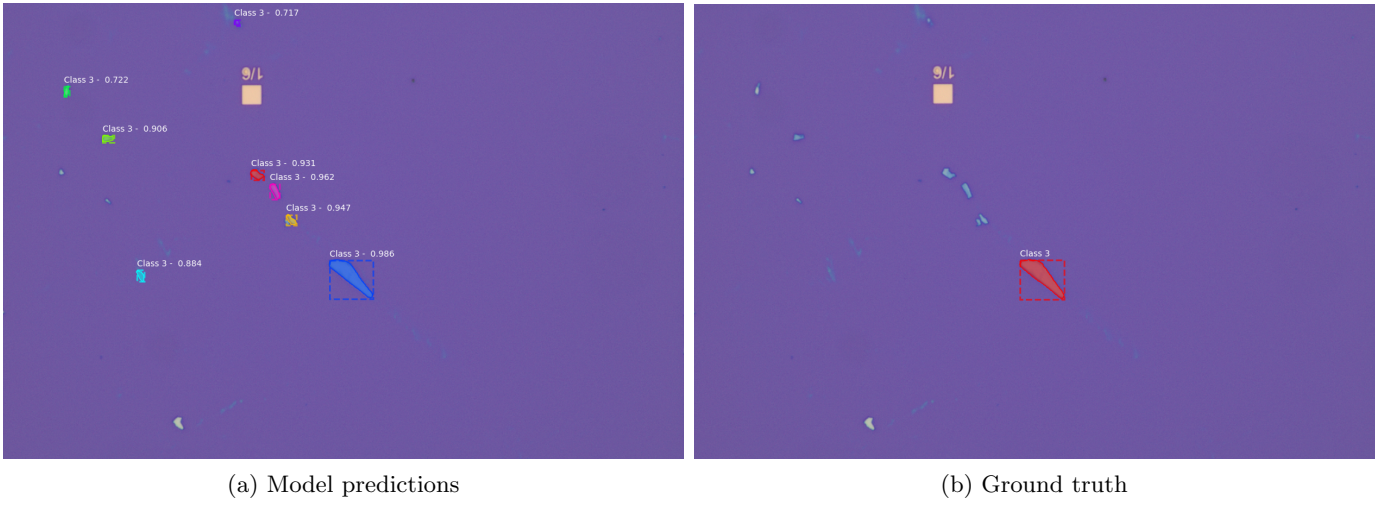


FIG. 29: Model predictions (a) and ground truth (b) of a test image. The value next to the class name in the predictions is the confidence value of the model.