MATH 6380P Advanced Topics in Deep Learning: Nexperia Kaggle Challenge

Andrea Madotto **Genta Indra Winata** **Zhaojiang Lin** **Jamin Shin**

Center of AI Research (CAiRE) Lab
{amadotto,giwinata,zlinao,jmshinaa}@connect.ust.hk

Abstract

Fault detection for semiconductors is an important manufacturing stage to distinguish good and bad components, especially for one of the biggest semiconductor company in the world, Nexperia[1]. To automatically solve this problem, we benchmark several classifiers, both feature based and fully differentiable models, and we tried a recently proposed Differentiable Architecture Search [1] which attempts to automatically design high-performance convolutional architectures for image classification. We analyze the obtained results using both t-SNE on the extracted features and Gradient-weighted Class Activation Mapping (Grad-CAM) [2]. Our results show that DARTS, which is trained from scratch, achieve the best AUC of 99.06 in the test set, that is also close to other transferring learning solution (Res-Net finetuning).

1 Dataset

The dataset is made of two classes, one for bad semi-conductor and another for good. This challenge aims to predict the quality of each semiconductor based on the image. The dataset has 30K and 3000 images for training and testing, respectively. We further split the training set to obtain a validation set of 300 images (150 bad and 150 good).

2 Classifiers

We report four learning solutions: 1) feature-based classifiers, 2) transferring learning, 3) training from scratch a convolutional neural network (CNN) [3] model, 4) differentiable architecture search method (DARTS).

Feature-based classifiers We tried several classifiers such as LDA, Random Forest (RM), Logistic Regression (LR) and Support Vector Machine (SVM) [4] using the penultimate layer of Res-net 18 model as features (512-dimensional vector for each image). We use scikit-learn for the classifiers and Torchvision of Pytorch library for the pre-trained model.

Transfer learning We use Res-Net 18 with the last layer redefined to match our binary classification problem. Then, we fine-tuned this new model, both by fully back-propagating the error (Res-Net18 FBP) and by just training the last layer (Res-Net18 LL).

Train from scratch We train a two-layer convolutional neural network with a random initialization (no fine tuning), and where each layer has a two-dimensional convolutional filter with ReLU activation

[1] https://www.nexperia.com/

Preprint. Work in progress.

Algorithm 1: DARTS – Differentiable Architecture Search

Create a mixed operation $\bar{o}^{(i,j)}$ parametrized by $\alpha^{(i,j)}$ for each edge (i,j)
while *not converged* **do**
1. Update weights w by descending $\nabla_w \mathcal{L}_{train}(w,\alpha)$
2. Update architecture α by descending $\nabla_\alpha \mathcal{L}_{val}(w - \xi\nabla_w \mathcal{L}_{train}(w,\alpha),\alpha)$

Replace $\bar{o}^{(i,j)}$ with $o^{(i,j)} = \text{argmax}_{o\in\mathcal{O}}\, \alpha_o^{(i,j)}$ for each edge (i,j)

Figure 1: DARTS algorithm [1]

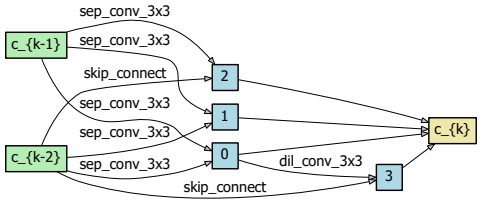


Figure 2: Normal cell learned on ImageNet

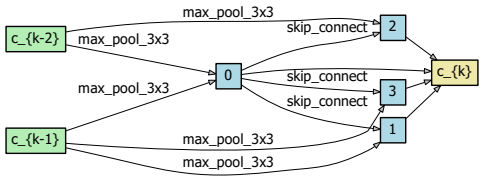


Figure 3: Reduction cell learned on ImageNet

function and max pooling. The first layer has 16 channel outputs with a kernel size of 5 and the second layer has 32 channels with 0.5 dropout. Finally, we use two fully connected (1000 hidden state) layers for the final binary prediction.

Differentiable ARchiTecture Search (DARTS) is a method to efficiently search the best model architecture by relaxing the discrete search space to continuous space [1]. The approach uses a gradient-based optimization that takes significantly less time than manual searching, and they represent the search as a directed acyclic graph. The building blocks consist of ordered N nodes, where each node $x^{(i)}$ is a latent representation and each directed edge (i,j) $o^{(i,j)}$ is multiplied with $x^{(j)}$ to transform $x^{(i)}$ as following:

$$x^i = \sum_{j<i} o^{(i,j)}(x^{(j)}) \tag{1}$$

During the search, the algorithm decides which connections should be kept or removed from the architecture. The loss is optimized by calculating the loss of the validation set with subject to training loss as a bi-level optimization problem. Figure 3 shows the algorithm for training and extract the final architecture, for more details refer to [1]. We run the search using Nexperia dataset, and we also take the DARTS architecture trained using ImageNet [5], which was provided by the authors [1]. The search network consists of a stack of eight network cell, the found cell are show in Figure 2,3,4,5. In this network, we did not use any pretrained feature extractor, and we train DARTS search using the following hyper-parameters:

Table 1: Hyper-parameters for DARTS

Parameters	**Value**
batch size	32
init lr	0.1
momentum	0.9
weight decay	3e-5
init channels	48
layers	14
grad clip	5
label smooth	0.1
gamma	0.97

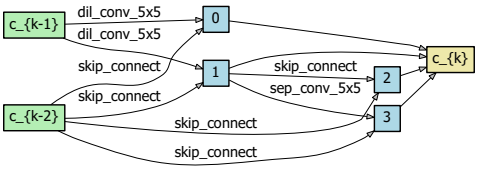


Figure 4: Normal cell learned on Nexperia

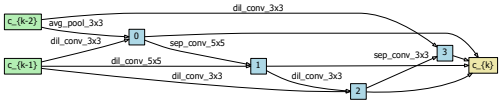


Figure 5: Reduction cell learned on Nexperia

3 Results

Table 2 summarizes the results of our experiments, and we test the model which achieved a high validation accuracy. The first thing to notice is that feature based classifiers, including *Res-Net LL*, do not achieve an accuracy, never greater than 60%, in the validation set. When we use *Res-Net FBP* feature, we can achieve good results; this is also confirmed by the t-SNE visualization shown in the analysis section. Also, a standard two layers *CNN* learned from scratch has a low accuracy (77%)[2].

Then we can see that fine-tuning Res-net gives a better result, highest validation accuracy (97.33 %), which shows the importance of using pre-trained model. Finally, DARTS, both search over ImageNet and from Nexperia, give a high validation accuracy and the best AUC in the test set. Importantly, the architectures found from DARTS are learned from scratch, which shows the effectiveness of the discovered model's inductive bias.

Table 2: Results for Nexperia dataset. In parentesis the accuracy using finetuned features from Res-Net FBP.

Model	Valid Acc	Test AUC
LDA	58.0 (96.0)	-
RF	54.0 (95.6)	-
LR	55.3 (95.6)	-
SVM	57.0 (95.6)	-
Res-Net LL	60.67	-
2-layer CNN	77.0	-
Res-Net FBP	**97.33**	98.30
DARTS ImageNet	95.33	**99.06**
DARTS Nexperia	95.67	98.57

4 Analysis

In this section, we visualize the Res-Net feature extractor using Grad-CAM [2] and t-SNE.

4.1 Grad-CAM

We use Grad-CAM to visualize the regions of input that are important for predictions. Here, we briefly describe the approach in [2]:

Given an image and a class (e.g. 'good') as input, we forward propagate the image through the model to obtain the raw class scores before the softmax layer. The gradients are set to zero for all classes except the desired one (e.g. 'good' or 'bad'), which is set to 1. This signal is then backpropagated to the rectified convolutional feature map of interest, where we can compute the coarse Grad-CAM localization (blue color).

The penultimate layer produces K feature maps $A^k \in \mathbf{R}^{u \times v}$ with width u and height v. These feature maps are then spatially pooled using Global Average Pooling (GAP) and linearly transformed

[2]We did not finetune the model at best, but we consider it as a baseline for other models.

to produce a score y^c for each class c. In order to obtain the class-discriminative localization map Grad-CAM $L^c_{Grad-CAM} \in \mathbf{R}^{u \times v}$, we first compute the gradient of y^c with respect to feature maps A of a convolutional layer, i.e. $\frac{\partial y^c}{\partial A^k_{ij}}$. The gradient is global-average-pooled to obtain weights α^c_k:

$$\alpha^c_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A^k_{ij}} \tag{2}$$

Grad-CAM heat-map is a weighted combination of feature maps and they apply ReLU to the sum.

$$L^c_{Grad-CAM} = \text{ReLU}(\sum_k \alpha^c_k A^k) \tag{3}$$

We show one good example on Figure 6 and one bad sample on Figure 9 from the training set. As we can see from Figure 7, when we input a correct category (good) for the good sample, the gradient is higher on the chip center. While in Figure 8 the gradient higher outside the chip when we input a wrong category (bad) for the same sample. In contrast, given a bad sample, Figure 10 shows the model focus on the damaged part of the chip, Figure 11 shows the model focus on the good part of the chip.

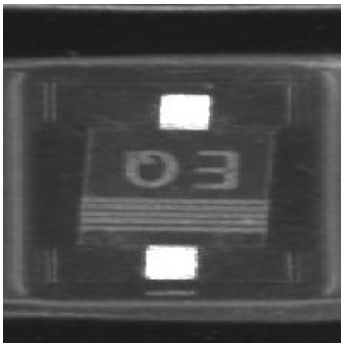


Figure 6: Sample labeled as good in the training set

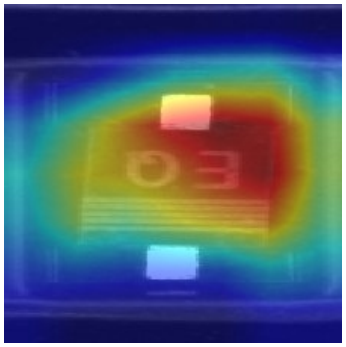


Figure 7: After backpropagation using correct labels

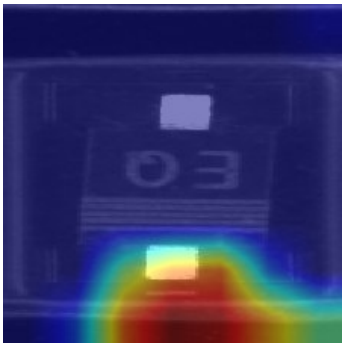


Figure 8: After backpropagation using wrong labels

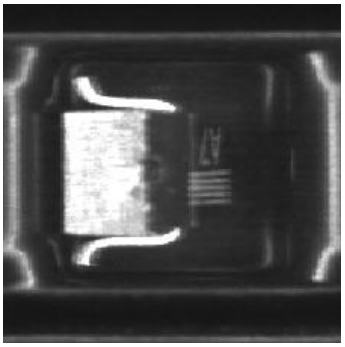


Figure 9: Sample labeled as bad in the training set

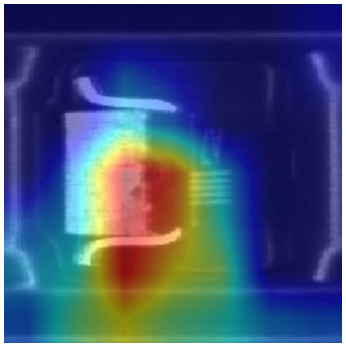


Figure 10: After backpropagation using correct labels

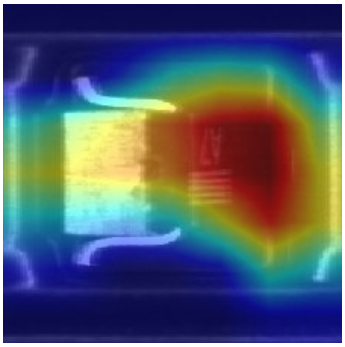
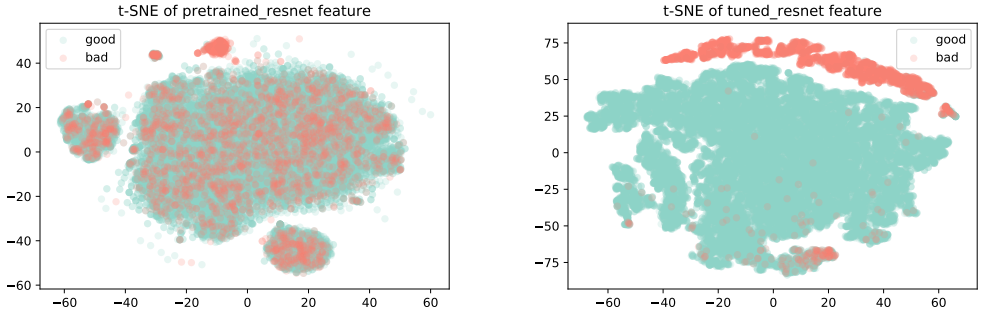


Figure 11: After backpropagation using wrong labels

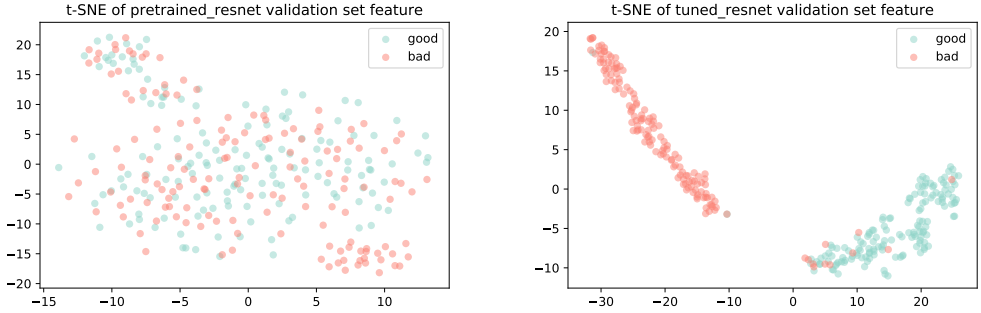
4.2 t-SNE

The features extracted by the pre-trained Res-Net are still mixed, while after fine tuning Res-Net on Nexperia, the features become well separated. Figure 13 shows such feature using t-SNE.



(a) Features of the pretrained Res-Net (b) Features of fine-tuned Res-Net

Figure 12: t-SNE training set visualization



(a) Features of the pretrained Res-Net (b) Features of fine-tuned Res-Net

Figure 13: t-SNE validation set visualization

References

[1] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[2] RS Ramprasaath, D Abhishek, V Ramakrishna, C Michael, P Devi, and B Dhruv. Grad-cam: why did you say that? visual explanations from deep networks via gradient-based localization. *CVPR 2016*, 2016.

[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[4] Vladimir Vapnik and Sayan Mukherjee. Support vector method for multivariate density estimation. In *Advances in neural information processing systems*, pages 659–665, 2000.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. Ieee, 2009.