Universitat Oberta de Catalunya

Image source: International Telecommunication Union (ITU) is the United Nations, https://www.itu.int/

Machine Learning based scratches on printed paper detection, in high-speed printing systems

Universitat Oberta de Catalunya
Màster Universitari en Enginyeria Informàtica
Treball Final de Màster - Intel·ligència Artificial

Professor responsable de l'assignatura: Carles Ventura Royo
Consultor: Antonio Burguera Burguera
Alumne: Jordi Falcés i Valls

Idioma: Anglès

Gener de 2021

The student

Jordi Falcés i Valls

Bachelor's Degree
Computer Science Engineering

Master's Degree
Computer Science Engineering

Universitat Oberta de Catalunya

Customer Assurance Master Engineer
HP PageWide Industrial

Agenda

- The idea
- The approach and method
- State-of-the-art research
- Creation of the datasets
- The experiments
- Conclusions
- Limitations
- Summary

The idea

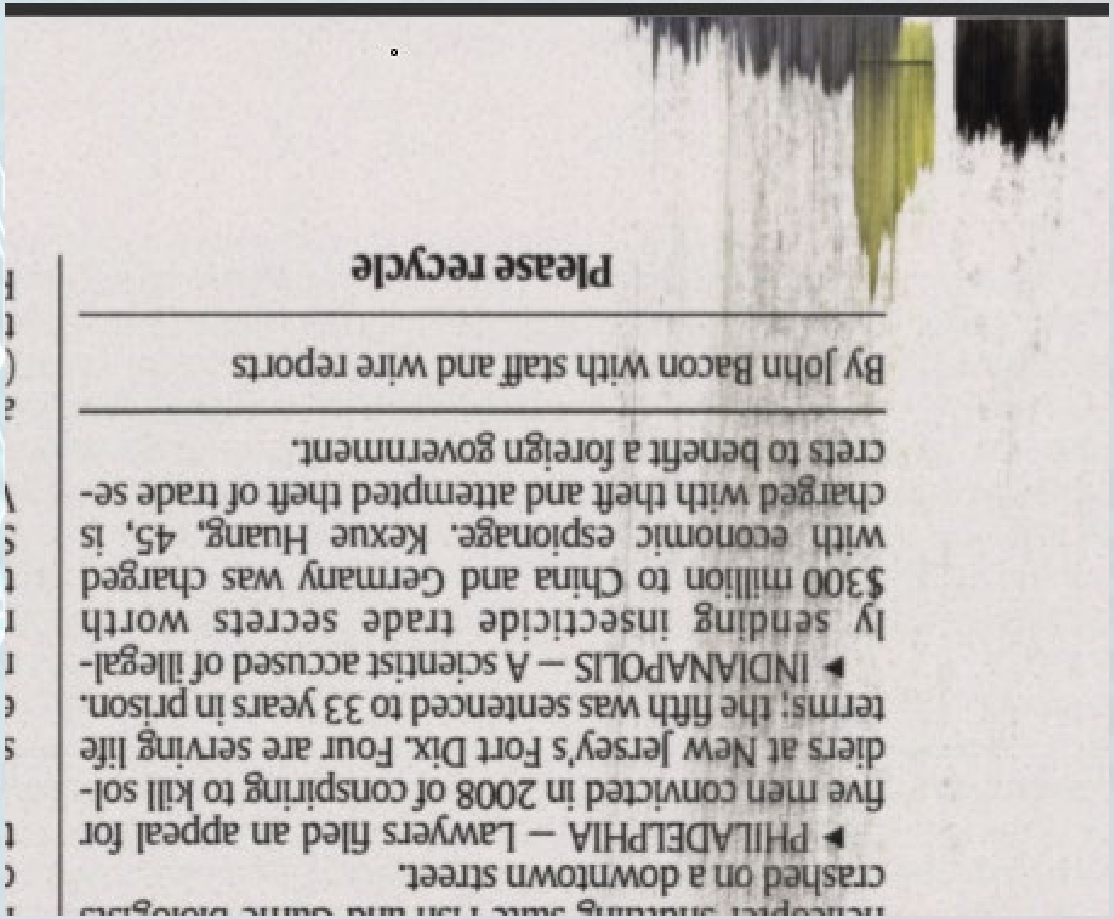
- The first idea

Machine Learning based <u>defect</u> on printed paper detection, in high-speed printing systems

Missing nozzles, bleeding, misregistration, spray, scratches, ghosting, picking, offsetting, wrinkling, etc.

- The complexity
- The final idea

Machine Learning based <u>scratches</u> on printed paper detection, in high-speed printing systems



The approach and method

Phase 1
State-of-the-art research

↓

Phase 2
Creation of the datasets

↓

Phase 3
The experiments

State-of-the-art research

- **Print quality** and **reliability** are more and more **demanding** over time.
- Defects in printed matter may cause **customer complaints**.
- Defects in printed matter may require **complete reprint**.
- Print shops want to avoid printing material waste and **look for increased profit margin**.
- Human inspection requires **dedicated operators** per printer.
- Human inspection **accuracy fluctuates**, defects are overlooked and speed is limited.
- Some applications may require **100% inspection rate**, which is not possible at high-speeds.

- **Automation** is a must for defect detection and classification.

State of the art research

- **Deep learning** has been successfully applied to classification tasks in many fields due to its good performance in learning discriminative features but the application to **printing defect classification** is **very rare**.
- **Pre-processing** may be required to remove noise, remove scanning or camera artifacts, blurring, etc.
- The kind of defect has to be considered for its own **characteristics**:
  - **SCRATCHES** are difficult to detect with general purpose methods. Specific Scratch Detector may be required.

State of the art research

- **Small** and **imbalanced datasets** is a problem.
  - **Augmentation**.
  - Oversampling.
  - **Undersampling**.
  - Synthetic Sampling with Data Generation.
  - Pre-train networks and transfer learning to avoid overfitting.
- **Real-time** (due to high-speed printing) is a problem.
  - Use model weighting and model pruning techniques.
  - Using **GPU** (or FPGA) instead of CPU can help with real-time (or very fast) requirements.

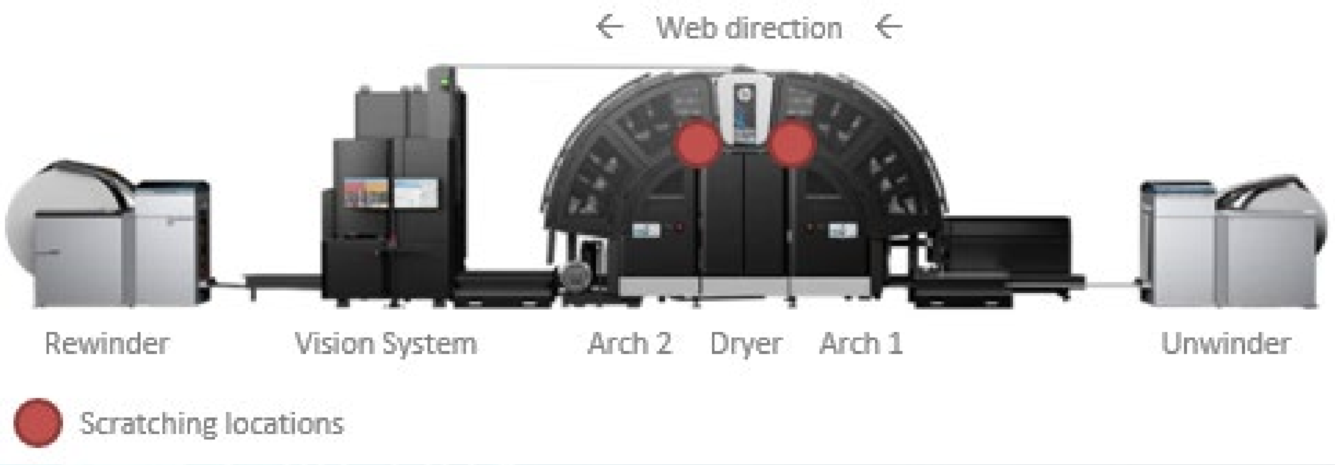
Creation of the datasets

1. **Defect generation** (RGB)
2. **Tiling** (64x64px and 320x320px)
3. **Conversion to Grayscale**
4. **Manual human selection**
5. **Balancing** (Undersampling)
6. **Augmentation**

Creation of the datasets

Defect generation

(RGB)



Creation of the datasets

Defect generation

(RGB)


Auto safety
Kid-in-car warning systems urged
Tech stocks slink back into do
Hot sector cools off as slowdown worries grow
Technology
Logitech sees a future in tel
Revue set-top box, Google TV bring Internet to your set
Facebook gift cards and Target are now
Retail giant plans to sell Credits cards for social network's games


Jessica Alba puts her game face on

With a new movie, *Machete*, in theaters Friday, the actress opens up about beefing up her roles, battling Latina stereotypes and being a mother, 3D

ntrol of care

Doctors' orders? Many begin to ask why

By Mary Brophy Marcus
USA TODAY

In the past, most patients placed their entire trust in the hands of their physician. Your doc said you needed a certain medical test, you got it.

Not so much anymore.

Jeff Chappell of Montgomery, Ala., recalls a visit a couple of years ago to a Charlotte emergency room, near where the family used to live, with his wife, Jacqueline, who has adrenal failure.

"I blew up loud enough for everyone in the ER to hear me explain that while we were insured, an MRI was about a $1,000 co-pay," Chappell says.

The couple knew her symptoms well (primarily stomach pain), knew that an MRI was not necessary under the circumstances and knew that a cortisone shot was what she needed.

"The doctor walked off in a huff," Chappell says, but later came back and "compromised" by agreeing to give his wife the shot, but not before taking an abdominal X-ray to rule out other problems first.

Cover story

Creation of the datasets

Tiling
(64x64px and 320x320px)

Creation of the datasets

Conversion to Grayscale

Creation of the datasets

Manual human selection

Creation of the datasets

Balancing
(Undersampling)

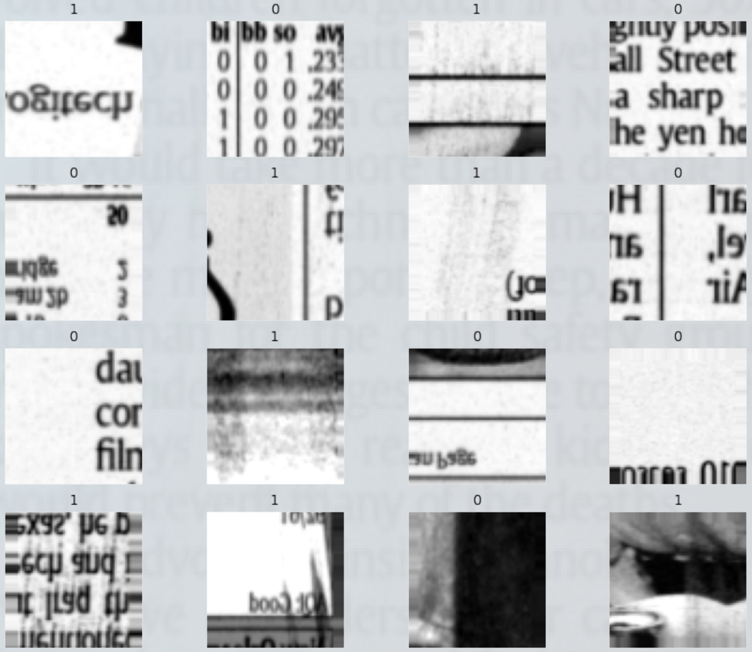
Scratches							
64x64px				320x320px			
Grayscale		Color		Grayscale		Color	
Scratched	Not-Scratched	Scratched	Not-Scratched	Scratched	Not-Scratched	Scratched	Not-Scratched
1,222	23,754	1,222	23,754	114	822	114	822
24,976		24,976		936		936	
49,952				1,872			
81,824							

Scratches							
64x64px				320x320px			
Grayscale		Color		Grayscale		Color	
Scratched	Not-Scratched	Scratched	Not-Scratched	Scratched	Not-Scratched	Scratched	Not-Scratched
1,222	1,222	1,222	1,222	114	114	114	114
2,444		2,444		228		228	
4,888				456			
5,344							

Creation of the datasets

Augmentation

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(
        rotation_range = 10, fill_mode = 'nearest', # Rotation
        width_shift_range = 0.2,                     # Horizontal shift
        height_shift_range = 0.2,                    # Vertical shift
        horizontal_flip = True,                      # Horizontal flip
        vertical_flip = True,                        # Vertical flip
        zoom_range = 0.2,                            # Zoom
        brightness_range = [0.2, 1.2])               # Brightness
```



The experiments

Experiment #	Tiles	Color/Grayscale	Balanced/Imbalanced	Augmentation
1	320x320px	Color	Balanced	No
2	320x320px	Color	Balanced	Yes
3	320x320px	Color	Imbalanced	No
4	320x320px	Color	Imbalanced	Yes
5	320x320px	Grayscale	Balanced	No
6	320x320px	Grayscale	Balanced	Yes
7	320x320px	Grayscale	Imbalanced	No
8	320x320px	Grayscale	Imbalanced	Yes
9	64x64px	Color	Balanced	No
10	64x64px	Color	Balanced	Yes
11	64x64px	Color	Imbalanced	No
12	64x64px	Color	Imbalanced	Yes
13	64x64px	Grayscale	Balanced	No
14	64x64px	Grayscale	Balanced	Yes
15	64x64px	Grayscale	Imbalanced	No
16	64x64px	Grayscale	Imbalanced	Yes

The experiments

```
# Configures and runs the model WITHOUT Augmentation
# GPU 37s

# Model architecture
model_exp1 = Sequential()

model_exp1.add(Conv2D(32, (3, 3), activation = 'relu',
                      input_shape = (320, 320, 3)))
model_exp1.add(MaxPooling2D(pool_size = (2, 2)))

model_exp1.add(Conv2D(32, (3, 3), activation = 'relu'))
model_exp1.add(MaxPooling2D(pool_size = (2, 2)))

model_exp1.add(Conv2D(32, (3, 3), activation = 'relu'))
model_exp1.add(MaxPooling2D(pool_size = (2, 2)))

model_exp1.add(Conv2D(64, (3, 3), activation = 'relu'))
model_exp1.add(MaxPooling2D(pool_size = (2, 2)))

model_exp1.add(Conv2D(64, (3, 3), activation = 'relu'))
model_exp1.add(MaxPooling2D(pool_size = (2, 2)))

model_exp1.add(Flatten())
model_exp1.add(Dense(64, activation = 'relu'))
model_exp1.add(Dropout(0.24))
model_exp1.add(Dense(2, activation = 'softmax'))

# Shows model summary
model_exp1.summary()

# Compiles the model
model_exp1.compile(loss = 'binary_crossentropy', metrics = ['accuracy'])

# Trains the model
history_exp1 = model_exp1.fit(x_train, y_train, epochs = 35, validation_data = (x_valid, y_valid), verbose = 1)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 318, 318, 32)      896
 max_pooling2d (MaxPooling2D)  (None, 159, 159, 32)    0
 conv2d_1 (Conv2D)           (None, 157, 157, 32)      9248
 max_pooling2d_1 (MaxPooling2D)  (None, 78, 78, 32)    0
 conv2d_2 (Conv2D)           (None, 76, 76, 32)        9248
 max_pooling2d_2 (MaxPooling2D)  (None, 38, 38, 32)    0
 conv2d_3 (Conv2D)           (None, 36, 36, 64)        18496
 max_pooling2d_3 (MaxPooling2D)  (None, 18, 18, 64)    0
 conv2d_4 (Conv2D)           (None, 16, 16, 64)        36928
 max_pooling2d_4 (MaxPooling2D)  (None, 8, 8, 64)      0
 flatten (Flatten)           (None, 4096)              0
 dense (Dense)               (None, 64)                262208
 dropout (Dropout)           (None, 64)                0
 dense_1 (Dense)             (None, 2)                 130
=================================================================
Total params: 337,154
Trainable params: 337,154
Non-trainable params: 0
_________________________________________________________________
```

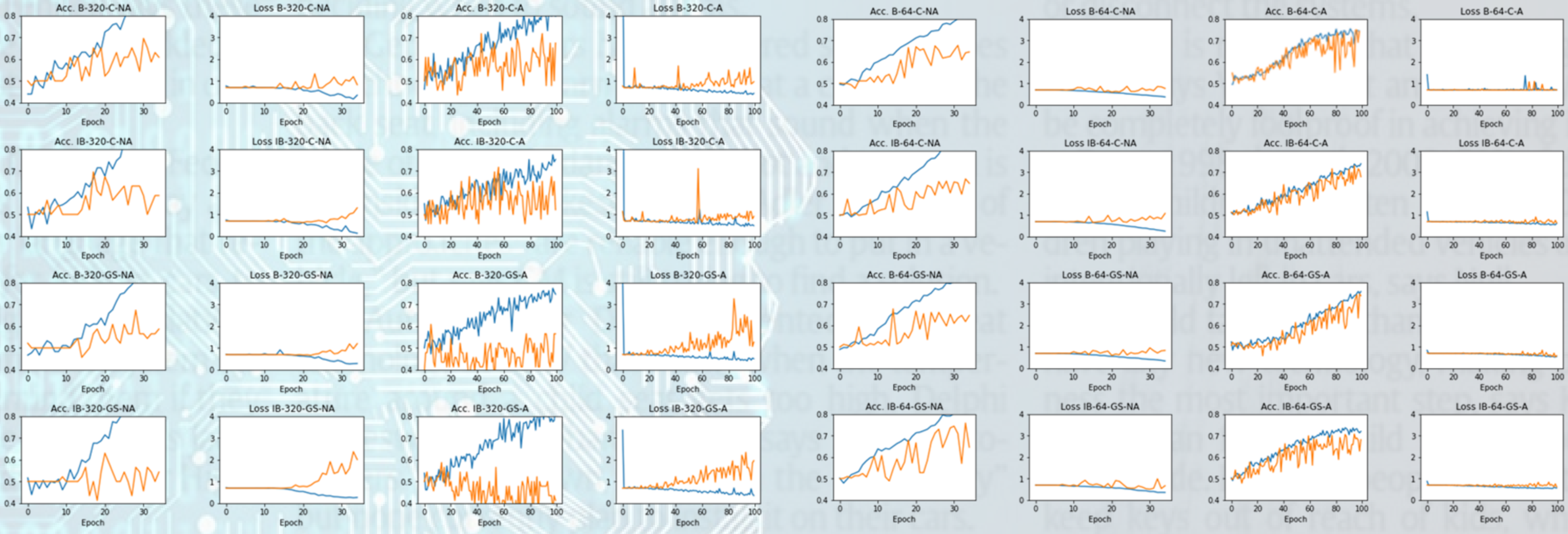
The experiments

```
# Configures and runs the model WITHOUT Augmentation
# GPU 33s

# Model architecture
model_exp9 = Sequential()

model_exp9.add(Conv2D(32, (3, 3), activation = 'relu',
                      input_shape = (64, 64, 3)))
model_exp9.add(MaxPooling2D(pool_size = (2, 2)))

model_exp9.add(Conv2D(32, (3, 3), activation = 'relu'))
model_exp9.add(MaxPooling2D(pool_size = (2, 2)))

model_exp9.add(Conv2D(32, (3, 3), activation = 'relu'))
model_exp9.add(MaxPooling2D(pool_size = (2, 2)))

model_exp9.add(Conv2D(64, (3, 3), activation = 'relu'))
model_exp9.add(MaxPooling2D(pool_size = (2, 2)))

model_exp9.add(Flatten())
model_exp9.add(Dense(64, activation = 'relu'))
model_exp9.add(Dropout(0.24))
model_exp9.add(Dense(2, activation = 'softmax'))

# Shows model summary
model_exp9.summary()

# Compiles the model
model_exp9.compile(loss = 'binary_crossentropy', metrics = ['accuracy'])

# Trains the model
history_exp9 = model_exp9.fit(x_train, y_train, epochs = 35, validation_data = (x_valid, y_valid), verbose = 1)
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_40 (Conv2D)          (None, 62, 62, 32)        896
 max_pooling2d_40 (MaxPooling2D) (None, 31, 31, 32)    0
 conv2d_41 (Conv2D)          (None, 29, 29, 32)        9248
 max_pooling2d_41 (MaxPooling2D) (None, 14, 14, 32)    0
 conv2d_42 (Conv2D)          (None, 12, 12, 32)        9248
 max_pooling2d_42 (MaxPooling2D) (None, 6, 6, 32)      0
 conv2d_43 (Conv2D)          (None, 4, 4, 64)          18496
 max_pooling2d_43 (MaxPooling2D) (None, 2, 2, 64)      0
 flatten_8 (Flatten)         (None, 256)               0
 dense_16 (Dense)            (None, 64)                16448
 dropout_8 (Dropout)         (None, 64)                0
 dense_17 (Dense)            (None, 2)                 130
=================================================================
Total params: 54,466
Trainable params: 54,466
Non-trainable params: 0
_________________________________________________________________
```

The experiments

Comparison summary of experiment charts



Train
Validation

B: Balanced IB: Imbalanced 320: 320x320px 64: 64x64px
C: Color GS: Grayscale NA: No Augmentation A: Augmentation

The experiments

Comparison summary of experiment table

Experiment	Dataset	Epochs at max. accuracy	Max. accuracy (%)	Execution time (s)
1	B-320-C-NA	20	65	37
2	B-320-C-A	50	65	600
3	IB-320-C-NA	18	70	33
4	IB-320-C-A	57	72	600
5	B-320-GS-NA	29	67	42
6	B-320-GS-A	44	72	180
7	IB-320-GS-NA	16	56	42
8	IB-320-GS-A	14	59	210
9	B-64-C-NA	5	48	33
10	B-64-C-A	65	72	420
11	IB-64-C-NA	16	63	42
12	IB-64-C-A	200	73	900
13	B-64-GS-NA	6	59	34
14	B-64-GS-A	100	75	240
15	IB-64-GS-NA	35	75	32
16	IB-64-GS-A	100	70	240

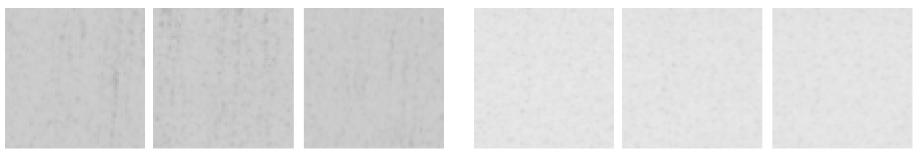
B: Balanced IB: Imbalanced 320: 320x320px 64: 64x64px C: Color GS: Grayscale NA: No Augmentation A: Augmentation

The experiments

- No significant difference between the results when using **imbalanced** or **balanced** datasets.
- No significant difference between the results when using **color** or **grayscale** datasets.
- Significant difference between the results when using **320x320px** and **64x64px** datasets.
  - Better results with 64x64px tiles datasets.
  - The model doesn’t do a good job when training using 320x320px tiles.
- Data **augmentation** has a positive impact in the training of the models.
- The **best-found dataset-training combination**: Experiment 14 (64x64px Grayscale Balanced Augmentation).
  - Accuracy of 79% after 400 Epochs, that requires 18 minutes of execution time using a Tesla K80 GPU in Google Colab.
  - Accuracy of 75% after 100 Epochs, that requires 4 minutes of execution time using a Tesla K80 GPU in Google Colab.

Conclusions

- **SCRATCHES** are **difficult to detect** with general purpose methods.
  - Scratches have special characteristics (very thin, very light contrast vs. background) that may require Specific Scratch Detector systems.
- The **difference** between a tile **with or without scratch** can be very subtle and can often be confused with noise in the image. This may also be hard for the machine learning system to detect.



Tiles with subtle scratch (LEFT) vs. images without scratch (RIGHT)

Conclusions

- It is **not easy** to create a **good dataset** from scratch.
  - Obtaining images that would be representative enough of the real world, image quality, quantity of elements, and balanced enough so it can be used in a machine learning system.
- Creating a good dataset is **time consuming** and, even part of the process can be automated (tiling, conversion to grayscale, etc.), there is still a classification that needs to be done by expert human eyes.

Conclusions

- Using **Undersampling** to create balanced datasets is a valid method but may remove important data that could potentially create a better dataset.
  - Undersampling is omitting information.
- Data **Augmentation** has a positive impact in the training of the models.
  - Resulted to be a valid method to increase the number of samples in the dataset.
  - The technique has to be designed accurately so no noise is introduced into the dataset.
  - The accuracy grows much more over epochs, even it requires more epochs to reach better accuracy.

Conclusions

- The **model requires more development** for datasets without augmentation.
  - Most of them show a divergence between the accuracy during training and the accuracy during validation, around Epochs 10 to 20.
- **Machine learning** has been tested as a solution to detect scratches in printed content, without needing to compare the printout with the original image.
  - Required a dataset created in-purpose and a model to be trained.
  - The accuracy has been found to be up to 75% to 79%, which is higher than the accuracy of 67% reported in previous studies using specific scratch detection systems.

Limitations

- An **accuracy** of 75% to 79% may not be enough for systems requiring high-precision.
- **Machine Learning** works as a **black box**.
    - It’s almost impossible to troubleshoot what rules have been applied to determine if a tile has or has not a scratch on it.
- Printing at **very-high-speeds** (up to 1000fpm in HP PageWide Web Presses) makes it unrealistic to capture every single printed frame (page), have it converted to grayscale, tiled, and verified by the trained machine learning model fast enough to report findings. If the application accepts sampling (analyze only a subset of captures), that would be enough but, for applications requiring **high level of inspections**, the solution may not work because of technology limitations (network bandwidth, processor, display, etc.

Summary

- The idea
- The approach and method
- State-of-the-art research
- Creation of the datasets
- The experiments
- Conclusions
- Limitations

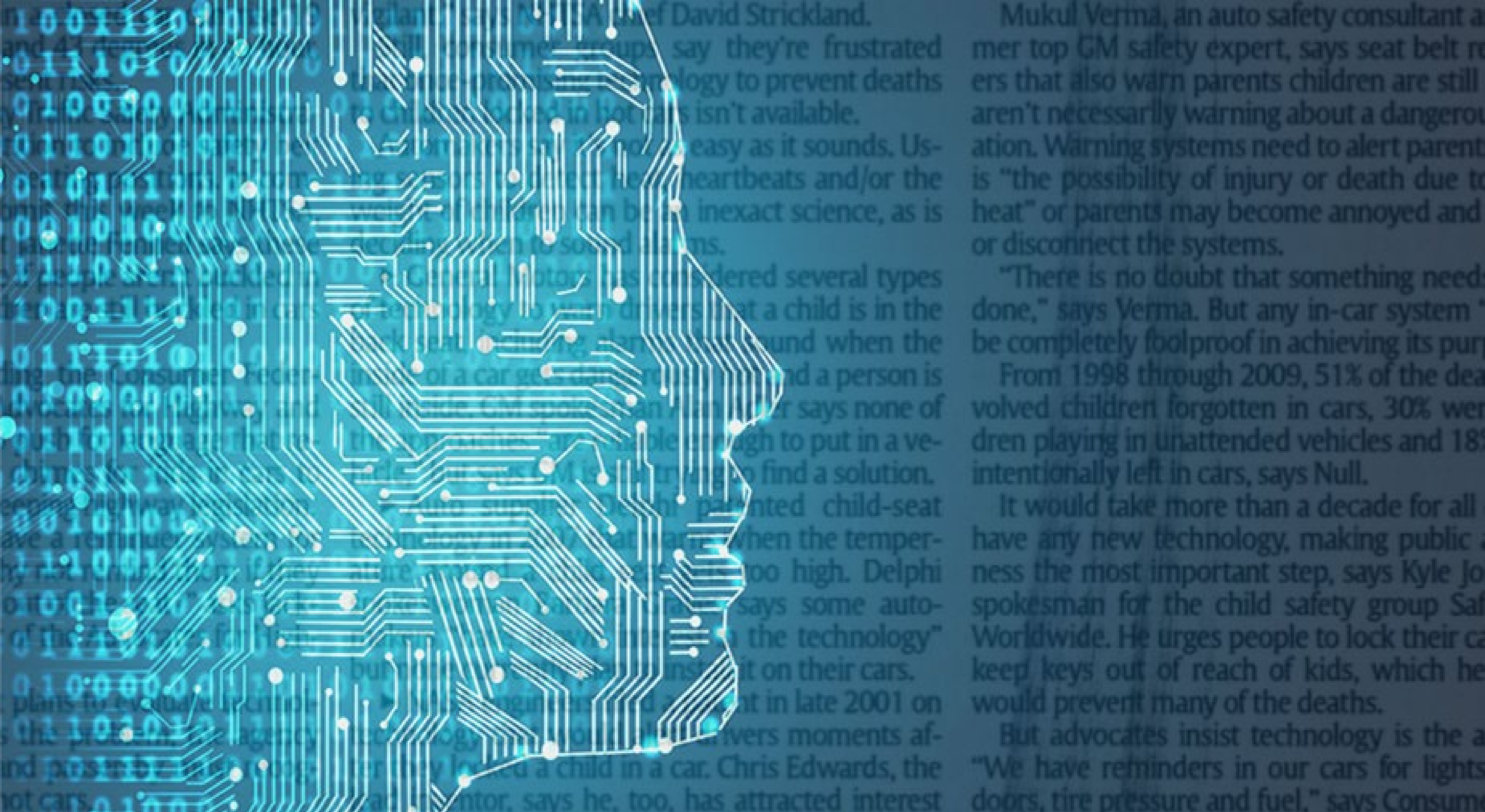


Image source: International Telecommunication Union (ITU) is the United Nations, https://www.itu.int/