Ансамблеві алгоритми є потужним інструментом у задачах класифікації, забезпечуючи високу ефективність моделей. Ці методи дозволяють ефективно комбінувати переваги різних підходів, роблячи моделі більш стійкими до похибок і варіативності даних. Існують різні підходи до ансамблевих методів, такі як averaging, resampling, cyclic learning rate алгоритми.

В цій курсовій ми розглянемо такі методи як: weighted (averaging частковий випадок цього алгоритму коли всім моделям ансамблю ми даємо рівномірно розподілені ймовірності), resampling, horizontal. Для кожного алгоритму ми будемо брати різні датасети, а саме: {weighted: cifar10, resampling: mnist,

horizontal: fashion\_mnist}

Також для тренування кожної моделі ми будемо використовувати аналогічні генератори, а саме:

|  |
| --- |
| **from** tensorflow**.***keras***.***preprocessing***.***image* **import** ImageDataGenerator  train\_datagen **=** ImageDataGenerator**(**rescale**=**1.**/**255**,**  rotation\_range **=** 15**,**  horizontal\_flip **=** **True,**  zoom\_range **=** 0.2**,**  shear\_range **=** 0.1**,**  fill\_mode **=** 'reflect'**,**  width\_shift\_range **=** 0.1**,**  height\_shift\_range **=** 0.1**)**  test\_datagen **=** ImageDataGenerator**(**rescale**=**1.**/**255**)**  train\_generator **=** train\_datagen**.***flow***(**train\_images**,** train\_labels\_ohe**,**  batch\_size **=** bat\_size  **)**  val\_generator **=** test\_datagen**.***flow***(**x\_val**,** y\_val**,**  batch\_size **=** bat\_size**,**  shuffle**=False**  **)**  test\_generator **=** test\_datagen**.***flow***(**x\_te**,** y\_te**,**  batch\_size **=** bat\_size**,**  shuffle**=False**  **)** |

Моделі для ансамблів в основному буду брати з [keras.applications](https://keras.io/api/applications/) ([ResNet101](https://keras.io/api/applications/resnet/#resnet101-function), [MobileNet](https://keras.io/api/applications/mobilenet/), [DenseNet121](https://keras.io/api/applications/densenet/#densenet121-function)).

|  |
| --- |
| **from** tensorflow**.***keras* **import** applications  resnet **=** applications**.***ResNet101***(**  include\_top**=True,**  weights**=** **None,**  input\_tensor**=None,**  input\_shape**=** train\_images**.***shape***[**1**:],**  classes**=**10**,**  classifier\_activation**=**'softmax'  **)**  mobilenet **=** applications**.***MobileNet***(**  input\_shape**=**train\_images**.***shape***[**1**:],**  alpha**=**1.0**,**  depth\_multiplier**=**1**,**  dropout**=**0.001**,**  include\_top**=True,**  weights**=** **None,**  input\_tensor**=None,**  pooling**=None,**  classes**=**10**,**  classifier\_activation**=**"softmax"**,**  **)**  desnet **=** applications**.***DenseNet121***(**  include\_top**=True,**  weights**=** **None,**  input\_tensor**=None,**  input\_shape**=** train\_images**.***shape***[**1**:],**  pooling**=None,**  classes**=**10**,**  classifier\_activation**=**"softmax"**,**  **)** |

**Weighted Ensemble**

Weighted Ensemble— це розширення підходу averaging моделей, де внесок кожного члена ансамблю у фінальний прогноз зважується відповідно до його продуктивності. Ваги моделей є малими додатними значеннями, причому їх сума дорівнює одиниці. Це дозволяє вагам вказувати на відсоток довіри або очікувану продуктивність кожної моделі.

Для перевірки цього алгоритму будемо ансамблювати (ResNet101, MobileNet та Model), де

Model:

|  |
| --- |
| **from** keras **import** Model**,** layers  net **=** **{}**  net**[**'input'**]** **=** layers**.***Input***((**32**,** 32**,** 3**))**  net**[**'conv\_1'**]** **=** layers**.***Conv2D***(**256**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'input'**])**  net**[**'batch\_1'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_1'**])**  net**[**'activ\_1'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_1'**])**  net**[**'conv\_2'**]** **=** layers**.***Conv2D***(**256**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'activ\_1'**])**  net**[**'batch\_2'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_2'**])**  net**[**'activ\_2'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_2'**])**  net**[**'pool\_1'**]** **=** layers**.***MaxPooling2D***(**pool\_size**=(**2**,**2**))(**net**[**'activ\_2'**])**  net**[**'dropout\_1'**]** **=** layers**.***Dropout***(**0.2**)(**net**[**'pool\_1'**])**  net**[**'conv\_3'**]** **=** layers**.***Conv2D***(**512**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'dropout\_1'**])**  net**[**'batch\_3'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_3'**])**  net**[**'activ\_3'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_3'**])**  net**[**'conv\_4'**]** **=** layers**.***Conv2D***(**512**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'activ\_3'**])**  net**[**'batch\_4'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_4'**])**  net**[**'activ\_4'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_4'**])**  net**[**'pool\_2'**]** **=** layers**.***MaxPooling2D***(**pool\_size**=(**2**,**2**))(**net**[**'activ\_4'**])**  net**[**'dropout\_2'**]** **=** layers**.***Dropout***(**0.2**)(**net**[**'pool\_2'**])**  net**[**'conv\_5'**]** **=** layers**.***Conv2D***(**512**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'dropout\_2'**])**  net**[**'batch\_5'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_5'**])**  net**[**'activ\_5'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_5'**])**  net**[**'conv\_5'**]** **=** layers**.***Conv2D***(**512**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'activ\_5'**])**  net**[**'batch\_6'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_5'**])**  net**[**'activ\_6'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_6'**])**  net**[**'pool\_3'**]** **=** layers**.***MaxPooling2D***(**pool\_size**=(**2**,**2**))(**net**[**'activ\_6'**])**  net**[**'dropout\_3'**]** **=** layers**.***Dropout***(**0.2**)(**net**[**'pool\_3'**])**  net**[**'conv\_7'**]** **=** layers**.***Conv2D***(**512**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'dropout\_3'**])**  net**[**'batch\_7'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_7'**])**  net**[**'activ\_7'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_7'**])**  net**[**'conv\_8'**]** **=** layers**.***Conv2D***(**512**,** kernel\_size**=(**3**,** 3**),** padding**=**'same'**)(**net**[**'activ\_7'**])**  net**[**'batch\_8'**]** **=** layers**.***BatchNormalization***()(**net**[**'conv\_8'**])**  net**[**'activ\_8'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'batch\_8'**])**  net**[**'pool\_4'**]** **=** layers**.***MaxPooling2D***(**pool\_size**=(**2**,**2**))(**net**[**'activ\_8'**])**  net**[**'dropout\_4'**]** **=** layers**.***Dropout***(**0.2**)(**net**[**'pool\_4'**])**  net**[**'flat'**]** **=** layers**.***Flatten***()(**net**[**'dropout\_4'**])**  net**[**'dense\_1'**]** **=** layers**.***Dense***(**1024**)(**net**[**'flat'**])**  net**[**'activ\_7'**]** **=** layers**.***Activation***(**'relu'**)(**net**[**'dense\_1'**])**  net**[**'dropout\_4'**]** **=** layers**.***Dropout***(**0.2**)(**net**[**'activ\_7'**])**  net**[**'batch\_7'**]** **=** layers**.***BatchNormalization***()(**net**[**'dropout\_4'**])**  net**[**'output'**]** **=** layers**.***Dense***(**10**,** activation**=**'softmax'**)(**net**[**'batch\_7'**])**  model **=** Model**(**net**[**'input'**],** net**[**'output'**])**  model**.compile(**loss**=**'categorical\_crossentropy'**,**  optimizer**=**'adam'**,**  metrics**=[**'acc'**])** |

Для weighted ensemble також напишемо шар, щоб задавати ваги моделям

|  |
| --- |
| **class** **WeightedAverageLayer(**tensorflow**.***keras***.***layers***.***Layer***):**  **def** \_\_init\_\_**(**self**,** w1**,** w2**,** w3**,** **\*\***kwargs**):**  **super(**WeightedAverageLayer**,** self**).***\_\_init\_\_***(\*\***kwargs**)**  self**.***w1* **=** w1  self**.***w2* **=** w2  self**.***w3* **=** w3  **def** call**(**self**,** inputs**):**  **return** self**.***w1* **\*** inputs**[**0**]** **+** self**.***w2* **\*** inputs**[**1**]** **+** self**.***w3* **\*** inputs**[**2**]** |

Отже спочатку зафіксуємо accuracy кожної моделі окремо.

|  |
| --- |
| loss, acc **=** resnet.evaluate(test\_generator)  print("resnet model, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **3s** 15ms/step - acc: 0.8190 - loss: 1.3334  resnet model, accuracy: 81.76% |

|  |
| --- |
| loss, acc **=** mobilenet.evaluate(test\_generator)  print("mobilenet model, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **1s** 4ms/step - acc: 0.8499 - loss: 0.4718  mobilenet model, accuracy: 85.01% |

|  |
| --- |
| loss, acc **=** model.evaluate(test\_generator)  print("Model, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **3s** 12ms/step - acc: 0.8652 - loss: 0.4033  Model, accuracy: 87.11% |

|  |  |
| --- | --- |
| **Model name** | **Accuracy (%)** |
| ResNet101 | 81.76% |
| MobileNet | 85.01% |
| model | 87.11% |

Для початку перевіримо наскільки покращить averaging ensemble.

|  |
| --- |
| **from** tensorflow**.***keras* **import** layers**,** Model  **import** itertools  models **=** **[**resnet**,** mobilenet**,** model**]**  model\_input **=** layers**.***Input***(**shape**=(**32**,** 32**,** 3**))**  model\_outputs **=** **[**model**(**model\_input**)** **for** model **in** models**]**  flattened\_outputs **=** model\_outputs  ensemble\_output **=** layers**.***Average***()(**flattened\_outputs**)**  ensemble\_model **=** Model**(**inputs**=**model\_input**,** outputs**=**ensemble\_output**)**  ensemble\_model**.compile(**loss**=**'categorical\_crossentropy'**,**  optimizer**=**'adam'**,**  metrics**=[**'acc'**])** |

|  |
| --- |
| loss, acc **=** ensemble\_model.evaluate(test\_generator)  print("Average\_ensemble, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **22s** 47ms/step - acc: 0.8851 - loss: 0.3495  Average\_ensemble, accuracy: 88.69% |

Отже Averaging ensemble видав accuracy = 88.69%

Тепер перевіримо Weighted ensemble де кожній моделі задамо свої ваги, а саме

|  |  |
| --- | --- |
| ResNet101 | 2/12 |
| MobileNet | 3/12 |
| model | 7/12 |
| Σ | 1 |

|  |
| --- |
| models **=** **[**resnet**,** mobilenet**,** model**]**  model\_input **=** layers**.***Input***(**shape**=(**32**,** 32**,** 3**))**  model\_outputs **=** **[**model**(**model\_input**)** **for** model **in** models**]**  flattened\_outputs **=** model\_outputs  ensemble\_output\_w **=** WeightedAverageLayer**(**2**/**12**,** 3**/**12**,** 7**/**12**)(**flattened\_outputs**)**  ensemble\_model\_w **=** Model**(**inputs**=**model\_input**,** outputs**=**ensemble\_output\_w**)**  ensemble\_model\_w**.compile(**loss**=**'categorical\_crossentropy'**,**  optimizer**=**'adam'**,**  metrics**=[**'acc'**])** |

|  |
| --- |
| loss, acc **=** ensemble\_model\_w.evaluate(test\_generator)  print("Weighted\_ensemble, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **22s** 48ms/step - acc: 0.8845 - loss: 0.3379  Weighted\_ensemble, accuracy: 88.87% |

Фінальна таблиця результатів:

|  |  |
| --- | --- |
| **Model name** | **Accuracy (%)** |
| ResNet101 | 81.76% |
| MobileNet | 85.01% |
| model | 87.11% |
| Averaging ensemble | 88.69% |
| Weighted ensemble | 88.87% |

Найкращий результат показав Weighted ensemble, а саме 88.87% що на 0.18% більше ніж Averaging ensemble або ж на 1.76% більше ніж одна звичайна модель

Весь файл: [average\_weighted\_ensemble\_cifar10](https://drive.google.com/file/d/1LgxJfEWMN_ECdUpvvm2dpUUfuyW8gcd-/view?usp=sharing)

**Resampling Ensemble**

Resampling Ensemble - це метод ансамблювання, який сприяє різноманітності між моделями шляхом застосування одного і того ж алгоритму навчання до різних наборів даних. Це досягається шляхом повторного випадкового вибору навчальних даних, які використовуються для тренування кожної нової моделі. Кілька моделей навчаються на трохи різних підходах до навчальних даних, що призводить до різних помилок. При їх комбінуванні це призводить до більш стабільних і кращих прогнозів.

Для перевірки цього алгоритму будемо ансамблювати (ResNet101, MobileNet та DenseNet121)

Для кожної моделі поділимо дані на непересічні рівнопотужні групи

|  |
| --- |
| test\_generator **=** test\_datagen**.***flow***(**x\_te**,** y\_te**,**  batch\_size **=** bat\_size**,**  shuffle**=False**  **)**  train\_generator\_resnet **=** train\_datagen**.***flow***(**train\_images**[:**20000**],** train\_labels\_ohe**[:**20000**],**  batch\_size **=** bat\_size  **)**  val\_generator\_resnet **=** test\_datagen**.***flow***(**x\_val**[:**1000**],** y\_val**[:**1000**],**  batch\_size **=** bat\_size**,**  shuffle**=False**  **)**  train\_generator\_desnet **=** train\_datagen**.***flow***(**train\_images**[**20000**:**40000**],** train\_labels\_ohe**[**20000**:**40000**],**  batch\_size **=** bat\_size  **)**  val\_generator\_desnet **=** test\_datagen**.***flow***(**x\_val**[**1000**:**2000**],** y\_val**[**1000**:**2000**],**  batch\_size **=** bat\_size**,**  shuffle**=False**  **)**  train\_generator\_mobilenet **=** train\_datagen**.***flow***(**train\_images**[**40000**:],** train\_labels\_ohe**[**40000**:],**  batch\_size **=** bat\_size  **)**  val\_generator\_mobilenet **=** test\_datagen**.***flow***(**x\_val**[**2000**:],** y\_val**[**2000**:],**  batch\_size **=** bat\_size**,**  shuffle**=False**  **)** |

Зафіксуємо спочатку accuracy для кожної моделі

|  |
| --- |
| loss, acc **=** resnet.evaluate(test\_generator)  print("resnet model, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **7s** 30ms/step - acc: 0.8901 - loss: 0.3562  resnet model, accuracy: 89.09% |

|  |
| --- |
| loss, acc **=** desnet.evaluate(test\_generator)  print("desnet model, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **12s** 54ms/step - acc: 0.9528 - loss: 0.1642  desnet model, accuracy: 95.24% |

|  |
| --- |
| loss, acc **=** mobilenet.evaluate(test\_generator)  print("mobilenet model, accuracy: {:5.2f}%".format(100 **\*** acc))  **219/219** ━━━━━━━━━━━━━━━━━━━━ **2s** 11ms/step - acc: 0.8591 - loss: 0.7300  mobilenet model, accuracy: 86.29% |

|  |  |
| --- | --- |
| **Model Name** | **Accuracy (%)** |
| ResNet101 | 89.09% |
| DenseNet121 | 95.24% |
| MobileNet | 86.29% |

Щоб ансамблювати використаємо шар з попереднього ансамблю, а саме

|  |
| --- |
| **class** **WeightedAverageLayer(**tensorflow**.***keras***.***layers***.***Layer***):**  **def** \_\_init\_\_**(**self**,** w1**,** w2**,** w3**,** **\*\***kwargs**):**  **super(**WeightedAverageLayer**,** self**).***\_\_init\_\_***(\*\***kwargs**)**  self**.***w1* **=** w1  self**.***w2* **=** w2  self**.***w3* **=** w3  **def** call**(**self**,** inputs**):**  **return** self**.***w1* **\*** inputs**[**0**]** **+** self**.***w2* **\*** inputs**[**1**]** **+** self**.***w3* **\*** inputs**[**2**]** |

|  |
| --- |
| **from** tensorflow**.***keras* **import** layers**,** Model  **import** itertools  models **=** **[**resnet**,** desnet**,** mobilenet**]**  model\_input **=** layers**.***Input***(**shape**=(**32**,** 32**,** 1**))**  model\_outputs **=** **[**model**(**model\_input**)** **for** model **in** models**]**  flattened\_outputs **=** **list(**model\_outputs**)**  ensemble\_output **=** layers**.***Average***()(**flattened\_outputs**)**  ensemble\_model **=** Model**(**inputs**=**model\_input**,** outputs**=**ensemble\_output**)**  ensemble\_model**.compile(**loss**=**'categorical\_crossentropy'**,**  optimizer**=**'adam'**,**  metrics**=[**'acc'**])** |

|  |
| --- |
| ensemble\_model.evaluate(test\_generator)  **219/219** ━━━━━━━━━━━━━━━━━━━━ **39s** 72ms/step - acc: 0.9679 - loss: 0.1902  Resampling\_ensemble accuracy: 96.90% |

|  |  |
| --- | --- |
| **Model Name** | **Accuracy (%)** |
| ResNet101 | 89.09% |
| DenseNet121 | 95.24% |
| MobileNet | 86.29% |
| Resampling\_ensemble | 96.90% |

Отже Resampling\_ensemble покращив accuracy на 1.66% відносно найкращої моделі (DenseNet121)

Весь файл: [mnist\_resempling\_ensemble](https://drive.google.com/file/d/1Y3CChIZFc2l6R4SPEFrLBmQ4d02xnHOz/view?usp=sharing)

**Horizontal Ensemble**

Horizontal Ensemble - це метод ансамблювання, який використовує кілька моделей, отриманих з кінця блоку епох під час тренування, для створення ансамблю. Він був розроблений спеціально для проблем прогнозування, де навчальний набір даних є відносно невеликим порівняно з кількістю прогнозів, що потрібні моделі. Це призводить до високої варіативності продуктивності моделі під час тренування, що ускладнює вибір кінцевої моделі або будь-якої моделі навіть у кінці процесу тренування через велику варіативність продуктивності.

Для перевірки цього алгоритму побудуємо просту щільну нейронну мережу (DNN)

|  |
| --- |
| **from** keras **import** Model**,** layers  net **=** **{}**  net**[**'input'**]** **=** layers**.***Input***((**28**,** 28**))**  net**[**'flat'**]** **=** layers**.***Flatten***()(**net**[**'input'**])**  net**[**'dense\_1'**]** **=** layers**.***Dense***(**256**,** activation **=** 'relu'**)(**net**[**'flat'**])**  net**[**'dense\_2'**]** **=** layers**.***Dense***(**128**,** activation **=** 'selu'**)(**net**[**'dense\_1'**])**  net**[**'dense\_3'**]** **=** layers**.***Dense***(**64**,** activation **=** 'relu'**)(**net**[**'dense\_2'**])**  net**[**'output'**]** **=** layers**.***Dense***(**10**,**activation **=** 'softmax'**)(**net**[**'dense\_3'**])**  model **=** Model**(**net**[**'input'**],** net**[**'output'**])**  model**.compile(**loss**=**'categorical\_crossentropy'**,**  optimizer**=**'adam'**,**  metrics**=[**'acc'**])** |

Код щоб після n-кроку(80) ми зберігали моделі:

|  |
| --- |
| n\_epochs**,** n\_save\_after **=** 100**,** 80  **for** i **in** **range(**n\_epochs**):**  model**.***fit***(**train\_images**,** train\_labels\_ohe**,** epochs**=**1**,** verbose**=**0**,** validation\_data**=(**x\_val**,** y\_val**))**  **print(**f'step : {i}' **)**  **if** i **>=** n\_save\_after**:**  **print(**f'step : {i}' **)**  model**.***save***(**'/kaggle/working/model\_' **+** **str(**i**)** **+** '.h5'**)**  **from** tensorflow**.***keras***.***models* **import** load\_model  **def** load\_all\_models**(**n\_start**,** n\_end**):**  all\_models **=** **list()**  **for** epoch **in** **range(**n\_start**,** n\_end**):**  filename **=** '/kaggle/working/model\_' **+** **str(**epoch**)** **+** '.h5'  model **=** load\_model**(**filename**)**  all\_models**.***append***(**model**)**  **print(**'>loaded %s' **%** filename**)**  **return** all\_models |

Завантажимо моделі

|  |
| --- |
| **from** tensorflow**.***keras***.***models* **import** load\_model  **def** load\_all\_models**(**n\_start**,** n\_end**):**  all\_models **=** **list()**  **for** epoch **in** **range(**n\_start**,** n\_end**):**  filename **=** '/kaggle/working/model\_' **+** **str(**epoch**)** **+** '.h5'  model **=** load\_model**(**filename**)**  all\_models**.***append***(**model**)**  **print(**'>loaded %s' **%** filename**)**  **return** all\_models  members **=** load\_all\_models**(**n\_save\_after**,** n\_epochs**)**  **print(**'Loaded %d models' **%** **len(**members**))**  members **=** **list(reversed(**members**))** |

Функції щоб оцінити accuracy ансамблю

|  |
| --- |
| **from** sklearn**.***metrics* **import** accuracy\_score  **from** keras**.***utils* **import** to\_categorical  **def** ensemble\_predictions**(**members**,** test\_images**):**  yhats **=** **[**model**.***predict***(**test\_images**)** **for** model **in** members**]**  yhats **=** np**.***array***(**yhats**)**  summed **=** np**.sum(**yhats**,** axis**=**0**)**  result **=** np**.***argmax***(**summed**,** axis**=**1**)**  **return** result  **def** evaluate\_n\_members**(**members**,** n\_members**,** test\_images**,** testy**):**  subset **=** members**[:**n\_members**]**  yhat **=** ensemble\_predictions**(**subset**,** test\_images**)**  **return** accuracy\_score**(**testy**,** yhat**)**  single\_scores**,** ensemble\_scores **=** **list(),** **list()** |

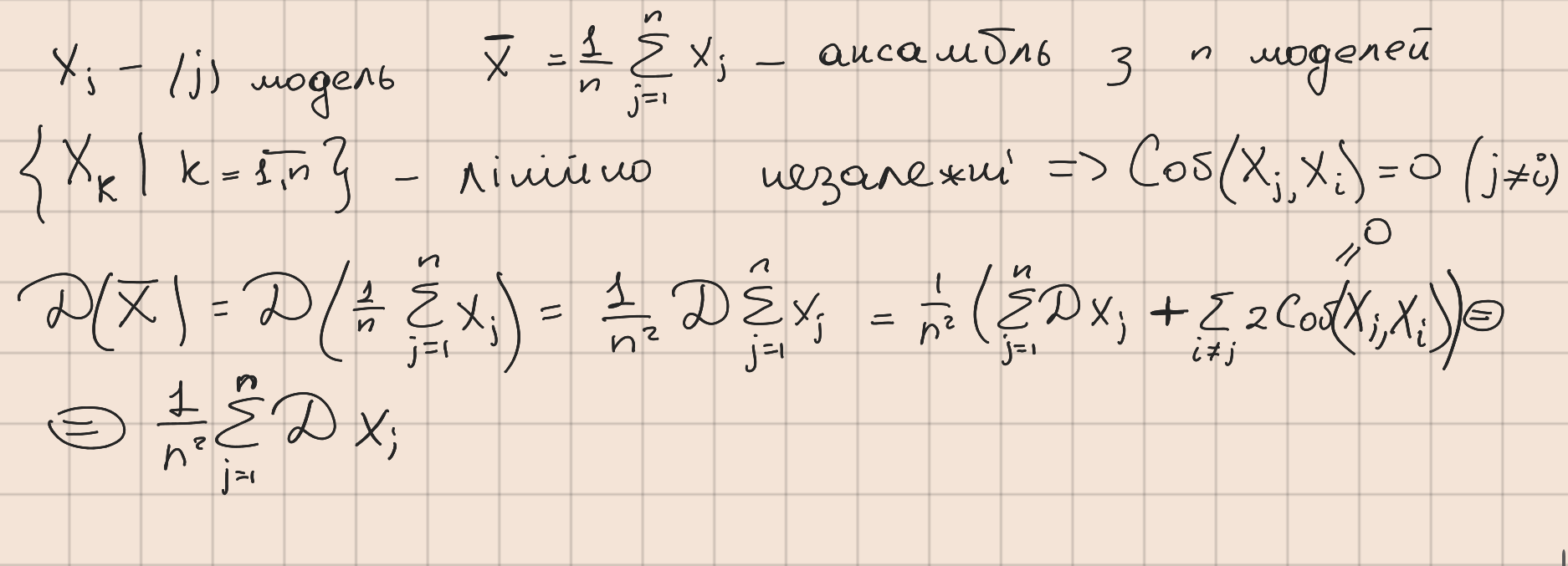
І сам алгоритм ансамблю

|  |
| --- |
| **for** i **in** **range(**1**,** **len(**members**)+**1**):**  ensemble\_score **=** evaluate\_n\_members**(**members**,** i**,** test\_images**,** test\_labels**)**  testy\_enc **=** to\_categorical**(**test\_labels**)**  \_**,** single\_score **=** members**[**i**-**1**].***evaluate***(**test\_images**,** testy\_enc**,** verbose**=**0**)**  **print(**'> %d: single=%.3f, ensemble=%.3f' **%** **(**i**,** single\_score**,** ensemble\_score**))**  ensemble\_scores**.***append***(**ensemble\_score**)**  single\_scores**.***append***(**single\_score**)** |

Результат:

|  |
| --- |
| Single score accuracy 87.3% Standard deviation = 0.014  Ensemble model accuracy 88.8% Standard deviation = 0.003 |

Результат є передбачуваним (зменшення дисперсії)



Отже ансамбль в цьому випадку покращив (accuracy, std) на (1.5%, 1.1%)

Весь файл: [horizontal\_ensemble\_fashion\_mnist](https://drive.google.com/file/d/16PGq5TIqrNZi7QxB1g6yFtCqGI3PXZg4/view?usp=sharing)

**Summary**

У цій курсовій роботі було досліджено ансамблеві алгоритми в задачах класифікації. Зокрема, були розглянуті методи averaging ensemble, weighted ensemble, resampling ensemble та horizontal ensemble. Проведені експерименти продемонстрували, що кожен з цих алгоритмів має свої переваги та може бути ефективним у різних ситуаціях.

Ансамблеві методи дозволяють ефективно комбінувати результати кількох моделей, що забезпечує підвищення точності та стійкості класифікації. Averaging ensemble є простим у реалізації та показує хороші результати для більшості задач. Weighted ensemble, у свою чергу, надає більше гнучкості, а саме можливість налаштовувати ваги окремих моделей, що дозволяє врахувати їхні індивідуальні особливості та продуктивність.

Алгоритм resampling дозволяє ефективно використовувати доступні дані шляхом повторного вибору підмножин навчальних даних. Це допомагає зменшити вплив дисбалансу класів, коли деякі класи представлені недостатньо, та підвищити загальну точність моделей. За рахунок створення нових вибірок даних та їх подальшого навчання, resampling підвищує стійкість моделі до змін у даних та покращує її здатність до узагальнення.

Horizontal ensemble, передбачає комбінування моделей на різних етапах обробки даних. Цей метод є особливо ефективним у задачах з великою кількістю змінних та складною структурою даних. Завдяки горизонтальному ансамблю можна врахувати різні аспекти даних та їх взаємозв'язки, що дозволяє побудувати більш точну та гнучку модель.

Таким чином, ансамблеві алгоритми є потужним інструментом для вирішення задач класифікації, забезпечуючи підвищення ефективності та гнучкісті моделі. Вибір конкретного методу залежить від специфіки задачі та характеристик даних, що обробляються.