УДК 636.084.74 DOI 10.31548/energiya6(82).2025.107

ВІЗУАЛЬНА ОЦІНКА СИРОВИНИ ПРИ ВИРОБНИЦТВІ ПРЕМІКСІВ З ВИКОРИСТАННЯМ OpenCV

М. О. Кіктєв, кандидат технічних наук, доцент

М. О. Правілов, аспірант

О. О. Опришко, кандидат технічних наук, доцент

О. М. Ромащук, кандидат технічних наук

Д. С. Лавінський, інженер

Національний університет біоресурсів і природокористування України

E-mail: m.pravilov@nubip.edu.ua

Анотація. *Контроль якості складових premіксів є важливим етапом їх виробництва. Від нього залежить низка показників кінцевого продукту (гомогенність, гранулометричні якості, ефективність складових premіксу тощо). У цьому дослідженні висвітлено доречність використання комп'ютерного зору на базі бібліотеки OpenCV для обробки зображень зразків солі та застосування глибокого навчання з допомогою TensorFlow для автоматизованої класифікації якості. За результатами дослідження виявлено, чи є доцільним використання наведених засобів для ефективного процесу контролю якості, що дає змогу зменшити вплив людського фактору та підвищити точність виявлення дефектів.*

Ключові слова: ***комп'ютерний зір, оцінка якості солі, виробництво premіксів, машинне навчання, OpenCV, TensorFlow.***

Актуальність. Основна мета додавання premіксів до комбікормів у відгодівлі тварин - це підвищення поживної цінності раціону та насичення його біологічно активними речовинами (вітамінами, мінералами, амінокислотами). Основні якісні показники цих сумішей залежить від якості вхідної сировини. Сіль є основним компонентом premіксів, бо є носієм мікроелементів і вітамінів. Від її якісних характеристик залежить рівномірність розподілу компонентів готового продукту.

Основними факторами, за якими оцінюють сіль перед використанням у технологічному процесі є: гранулометричний склад, комкуватість, вологість та наявність сторонніх домішок. Якщо за цими параметрами виявлено відхилення, то це може призвести до нерівномірного змішування компонентів, зниження ефективності готового продукту та фінансових втрат для виробників.

Наразі основним методом контролю якості солі є візуальна оцінка працівником виробництва, що є суб’єктивною та не завжди точною. З огляду на це, розробка автоматизованих методів оцінки якості солі є актуальною. Це забезпечило б швидке, точне та об’єктивне оцінювання вхідної сировини.

Застосування методів комп'ютерного зору та глибокого навчання є одним з можливих і перспективних напрямів у вирішенні цієї проблеми. Для цих цілей було використано бібліотеку для обробки зображень OpenCV та TensorFlow для навчання нейронних мереж. У перспективі це дасть змогу автоматизувати процес та мінімізувати людський фактор у процесі оцінки якості вхідної сировини.

Отже, розробка автоматизованої системи оцінки якості солі на основі аналізу зображень є актуальним завданням, яке відповідає сучасним тенденціям автоматизації та цифровізації сільського господарства, а саме виробництва кормів.

Аналіз останніх досліджень та публікацій. Одним із найважливіших компонентів комбікормів є збагачувальні та лікувальні мікродобавки (премікси). Організація їхнього виробництва на внутрішньогосподарських підприємствах є важливим резервом для підвищення якості та поживної цінності комбікормів. Нині відомо понад сто найменувань біологічно активних речовин різної хімічної природи та властивостей, що використовуються у виробництві премікcів. Розробляються рецептури премікcів для всіх виробничих груп тварин і птиці, в тому числі для автоматизованого виробництва. У роботі Victor Suresh, A. [1] наведено огляд програмного забезпечення для розрахунку рецептури премікcів. При адитивних технологіях, коли кінцевий продукт отримується з різноманітних базових компонентів за рахунок використання лінійного програмування та використанні математичних процедур, досягається отримання найкращого результату при наявності ряду обмежень.

Премікси, залежно від їхньої назви та вимог, поділяються на вітамінні (суміш вітамінних препаратів з наповнювачем), мінеральні (суміш мікроелементів з наповнювачем), комплексні (суміш багатьох компонентів з наповнювачем), лікувальні (лікарські препарати у профілактичних або лікувальних дозах), антистресові, ферментні та інші. Традиційний склад премікcів для великої рогатої

худоби майже всіх виробників включає вітаміни A, D, E, а також мікроелементи: цинк, мідь, марганець, кобальт, йод та селен. Цей набір біологічно активних речовин є дефіцитним у більшості регіонів нашої країни і, відповідно, премікси є необхідною умовою щодо правильної відгодівлі тварин, птиці та риби. Огляд останніх розробок та майбутніх для розробки стійкої формули корму для курей-несучок представлений у роботі Heidari, M.D. та ін. [2].

До недоліків преміксів можна віднести як обмежений термін зберігання, так і його порівняно високу вартість оскільки найбільш поширені рецепти преміксів передбачають використання високотехнологічних дорогих компонентів. Для зниження вартості проводяться розробки щодо заміщення імпортних місцевими ресурсами. Так, в роботі Wang, L.M. [3] приведено досвід заміни антибіотиків ефірними оліями та бензойною кислотою, а в дослідженнях Abdoun, A [4] запропоновано замінити змішані кормові раціони на базі люцерни листям *Moringa* spp. На території Східної та Центральної Європи також є місцеві ресурси придатні для створення преміксів, приклад розрахунку рецептури яких показано в роботі Кіктєва, М. [5].

Приклад використання цієї бібліотеки розглянуто в роботі Khort, D. та ін. [6] для розпізнавання координат полуниці при її збиранні роботом. Алгоритм автоматичного управління маніпулятором, реалізований об'єктно-орієнтованою мовою Python 3.7.2, включає операції з визначення координат X і Y ягоди, ступеня її зрілості, а також розрахунку відстані від маніпулятора до ягоди. Встановлено, що ефективність виявлення ягід, їх площі та кордонів із використанням камери та бібліотеки OpenCV при освітленості 300 Люкс досягає 94,6 відсотка. Освітлення істотно впливає на виявлення ягід суниці, їх площі, межі та стиглість за допомогою камери комп’ютерного зору.

Для автоматизації процесу оцінки якості солі використовувалася згорткова нейронна мережа (CNN), яка є ефективним інструментом для аналізу зображень. Процес навчання складався з кількох етапів, включаючи підготовку даних, вибір архітектури моделі та її оптимізацію.

Розпізнавання матеріалів за допомогою згорткових нейронних мереж описано в статті Bian, P. та ін. [7]. У цій статті пропонується новий підхід – ансамблеве навчання для розпізнавання матеріалів за допомогою згорткових нейронних мереж (CNN). У запропонованому методі модель CNN навчається для отримання ознак зображення, класифікатори на основі знань навчаються для отримання ймовірностей тестового зразка, що належить до різних категорій матеріалів. Автори пропонують три різні способи вивчення ознак ансамблю, що забезпечує більш високу точність розпізнавання. У статті Xu, H. та ін. [8] пропонується новий підхід до розпізнавання матеріалів, заснований як на трансферному навчанні, так і на моделі глибокої нейронної згорткової мережі (D-CNN). З цією метою автори створюють набір даних зображень сторонніх предметів, що складається із зображень із злітно-посадкових смуг міжнародного аеропорту Шанхай Хунцяо та кампусу науково-дослідного інституту. Автори оптимізують архітектуру D-CNN з огляду на характеристики розподілу матеріалів. Результати показують, що пропонований підхід може підвищити точність розпізнавання матеріалів на 39,6% порівняно із класичними існуючими методами.

При створенні таких високотехнологічних продуктів як премікси особливе значення має стабільність показників складових.

Мета дослідження – розроблення технологічних та програмно-технічних рішень для оцінки стану сировини безпосередньо на виробництві.

Матеріали та методи дослідження. Для обробки зображень використовувалася бібліотека OpenCV, яка забезпечує широкий спектр інструментів для аналізу. Основні етапи включають перетворення в градації сірого, згладжування, порогову обробку та аналіз контурів.

В Pycharm створені директорії з фото: train, validation і test. Для збільшення зразків для навчання зроблено аугментацію зображень. Далі обробку виконано за допомогою OpenCV, після чого проведено навчання фінальним кодом і тестування.

Результати досліджень та їх обговорення. *Eman 1 - Організація експерименту.* Збір зразків солі для дослідження проводився на декількох

виробництвах преміксу. Для аналізу використовувалися зразки солі з різних джерел, які класифікувалися за такими критеріями:

Комкувата сіль характеризується наявністю великих часток, що злиплися внаслідок підвищеної вологості;

Якісна сіль має рівномірний гранулометричний склад, відсутність комків.

Сіль різного помолу представляє собою дрібний та крупний помел.

Для відпрацювання методу був сформований датасет. Оригінальний набір фотографій з 25-30 зображень для якісної і 15-20 для неякісної солі, отриманих за різних умов освітлення та з різних ракурсів.. Після аугментації отримано по 700 для навчання нейронної мережі. Для навчання було використано 80 % зображень, решта 20 % – для тестування.

Етап 2 - Візуалізація зразків солі. Для аналізу були використані зразки солі різного помелу та якості, отримані від різних виробників. На рисунку 1 наведено приклади зразків солі з різними характеристиками, які були використані для навчання та тестування моделі.

На рис. 1 наведено якісну сіль дрібного помелу. Основними її характеристиками є: рівномірний розподіл частинок, дрібнодисперсна структура, відсутність домішок. Це забезпечує рівномірний розподіл солі в премісі, що впливає на стабільність хімічного складу готового продукту. Такий приклад сировини є найкращим для високоточних кормових добавок, де важливо уникати неоднорідності суміші.

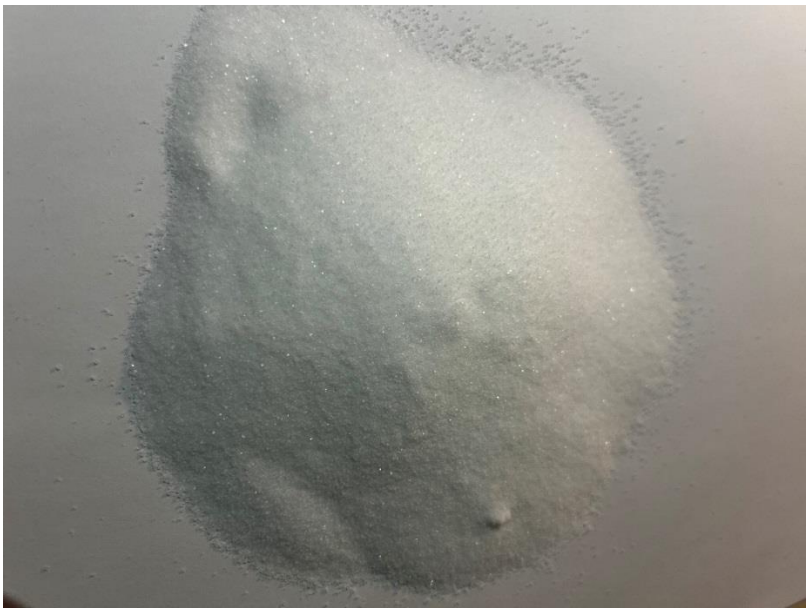

Рис.1. Якісна дрібнодисперсна сіль

На рис. 2, а представлено сіль крупного помелу та хорошої якості, з чітко вираженою гранулометричною структурою. Даний приклад сировини є допустимим, та має свої фізичні властивості, відмінні від зразка 1. На рис. 2, б показано крупну сіль з домішками, які можуть негативно впливати на якість кінцевого продукту. Домішками можуть бути різного походження: камінці, пил, бруд, зернові культури, які потрапили в сіль під час транспортування або видобування. Наявність домішок може викликати неоднорідність готової суміші та знижувати її ефективність.

На рис. 3 зображена неякісна сіль з комками, які утворилися внаслідок підвищеної вологості. Комкування солі є серйозною проблемою, оскільки призводить до зниження точності дозування та ускладнює технологічний процес виробництва премікcів. Висока вологість сприяє агрегації частинок, що ускладнює їх подальше рівномірне розподілення у складі кормових добавок.

Рис. 2. Якісна крупнодисперсна сіль (*а*), крупна сіль з домішками (*б*)

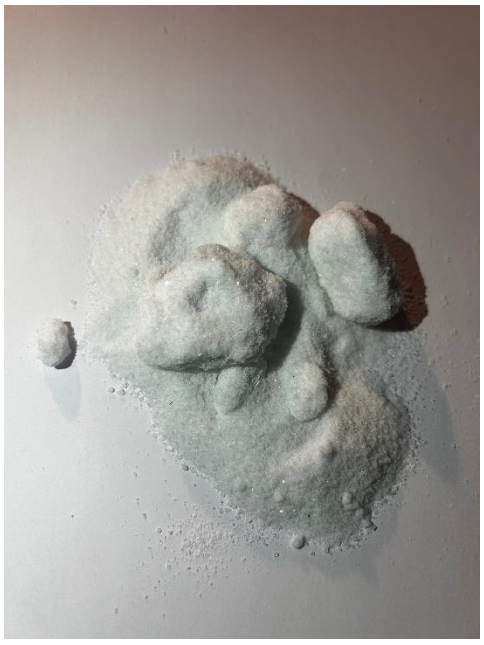

Рис. 3. Неякісна сіль з комками

Для коректності порівняння результатів аналізу всі зображення були отримані за однакових умов. Оскільки обробка зображень є важливим етапом перед оцінкою якості солі, то цей процес включає в себе декілька етапів. Перший етап - попередня обробка зображень. Вона включає в себе нормалізацію яскравості, корекцію контрасту та перетворення у градації сірого. Це дозволяє зменшити вплив зовнішніх факторів, таких як освітлення та тінь, на якість аналізу.

Наступним етапом є згладжування зображень за допомогою фільтрів. У нашому випадку це гаусове та середньозважене згладжування, які знижують рівень шуму та покращують виявлення країв. Далі виконується порогова обробка, яка дозволяє виділити ключові особливості зображення, а саме комки та сторонні включення.

Фінальним етапом є проведення аналізу контурів. Це дозволяє визначити найбільш важливі параметри оцінки якості солі - розмір і форму частинок. Використання адаптивних алгоритмів виявлення контурів дозволяє точно розділяти частинки солі та сторонні домішки. Цей аспект буде використовуватися в подальшій класифікації об’єктів.

На рис. А1 (додаток А) показано код програми на мові Python 3.13.2 для обробки зображень за допомогою OpenCV. На рис. 4 показано зображення неякісної солі після обробки.

Рис.4. Зображення неякісної солі після обробки за допомогою OpenCV

Навчання нейронної мережі. Першим кроком перед навчанням нейронної мережі було створення навчальних та тестових вибірок. Вони були заповнені зразками солі різного помолу та якості. Для збільшення кількості зразків було застосовано аугментацію (повороти, масштабування, зміна яскравості та ін.). Це дозволило зменшити ризик перенавчання. Архітектура нейронної мережі включала кілька згорткових шарів для автоматичного виділення ключових ознак зображення,

шари підсумовування для зменшення розмірності даних та повнозв’язні шари для остаточної класифікації.

Для навчання нейронної мережі використовувалася бібліотека TensorFlow. Архітектура моделі включала кілька згорткових шарів, шари підсумовування та повнозв'язні шари для остаточної класифікації. На початковому етапі проводиться імпорт необхідних бібліотек, що забезпечують роботу з даними та побудову нейронної мережі (таблиця).

1. Підготовка середовища та завантаження бібліотек

№ з/п	Команда	Призначення
1	**os**	Робота з файловою системою (створення, видалення та маніпулювання директоріями).
2	**numpy**	Чисельна бібліотека для роботи з масивами даних
3	**matplotlib.pyplot**	Побудова графіків навчання та візуалізація результатів
4	**tensorflow.keras**	Основний фреймворк для побудови та навчання нейронної мережі:
5	**ImageDataGenerator**, **load_img**, **img_to_array**	Завантаження та попередньої обробки зображень.
6	**Sequential**, **Input**, **Conv2D**, **MaxPooling2D**, **Flatten**, **Dense**, **Dropout**	Створення архітектури мережі.
7	**EarlyStopping**	Зупинка навчання у випадку, якщо модель починає перенавчатися.

Код для класифікації вхідних зображень зразків солі наведено у додатку Б. Він є основою для подальшої автоматизації процесу оцінки якості солі за допомогою методів комп'ютерного зору та глибокого навчання. Це дозволяє значно знизити вплив людського фактору та підвищити ефективність процесу змішування та дозування.

Результати навчання нейронної мережі. На представлених графіках відображено результати навчання моделі протягом п'яти епох. Графік точності показує, що модель демонструє стабільне зростання точності на навчальних даних:

від початкових 55 % до понад 83 % на п’ятій епосі. Це свідчить про ефективне засвоєння моделью особливостей тренувальної вибірки. Водночас точність на валідаційних даних коливається: після зростання до 72 % на третій епосі спостерігається спад до 57 % на четвертій, що може вказувати на початкові ознаки перенавчання або нерівномірність якості валідаційних даних. На п'ятій епосі точність знову підвищується до 72 %, що свідчить про певну стабілізацію.

Графік втрат підтверджує ці спостереження. Втрати на навчальних даних поступово знижуються з 1.2 до 0.43, що є типовою ознакою успішного навчання. Проте на валідаційних даних помітне зростання втрат на четвертій епосі до 1.05 після початкового зниження, що також може свідчити про потенційне перенавчання. На п'ятій епосі втрати на валідаційних даних зменшуються до 0.62, що вказує на можливе покращення узагальнюючої здатності моделі.

Таким чином, модель демонструє загальну позитивну динаміку в навчанні, однак спостерігається варіативність у точності та втратах на валідаційних даних. Це може свідчити про потребу в додатковій аугментації даних або коригуванні архітектури моделі для підвищення її стабільності та узагальнюючої здатності.

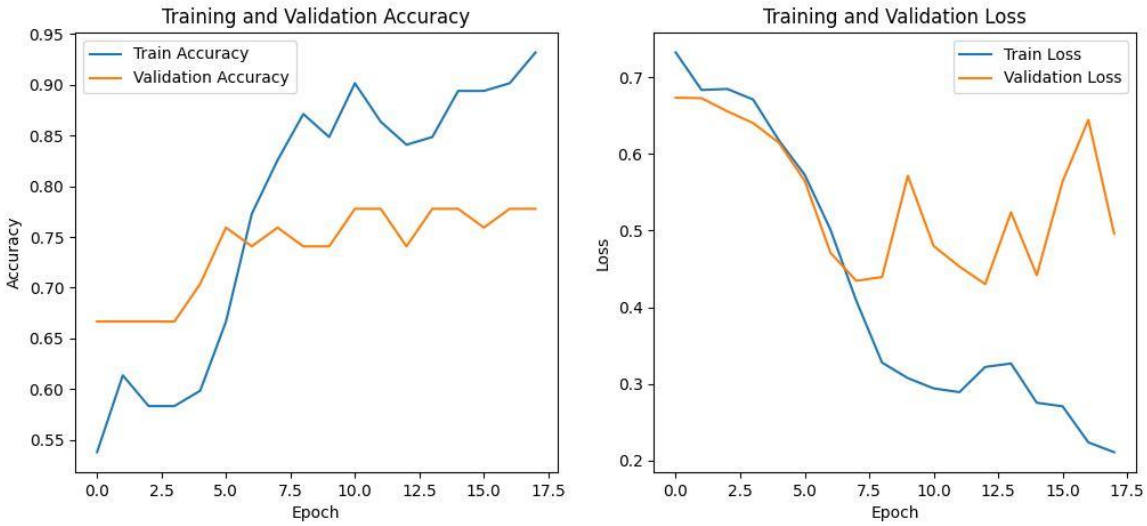

Рис.7. Результати навчання нейронної мережі в TensorFlow

Висновки і перспективи. У ході дослідження було розроблено програмний модуль для оцінки якості солі за допомогою методів комп'ютерного зору та

глибокого навчання на основі OpenCV і TensorFlow, який здатен аналізувати зображення зразків солі.

Розроблений модуль дозволяє оперативно виявляти основні проблеми якості солі: комкуватість та наявність сторонніх домішок. Завдяки цьому поліпшується точне дозування, рівномірність розподілу компонентів в преміксі, зменшується неоднорідність готової суміші.

Автоматизація процесу за допомогою машинного навчання сприяє зменшенню впливу людського фактору (помилки при візуальній оцінці), підвищенню точності виявлення дефектів, швидкості контролю якості, а також відкриває можливість інтеграції у виробничі лінії. Модуль дозволяє швидко приймати рішення про придатність вхідної сировини, забезпечуючи швидке, точне та об'єктивне оцінювання.

Подальші дослідження можуть бути зосереджені на розширенні набору даних для підвищення універсальності моделі, використанні складніших архітектур нейронних мереж, додаткової аугментації даних. З практичної сторони планується розробка і впровадження мобільних додатків для оперативного контролю якості на виробництві. Розроблений програмний модуль є основою для автоматизації процесу оцінки та має бути інтегрований у SCADA-систему контролю якості .Запропонована технологія здатна покращити якість премісів, що виробляються, і підвищити ефективність процесів на виробництві.

Список використаних джерел

1. A. Victor Suresh "Feed formulation software", Editor(s): Sergio F. Nates, Aquafeed Formulation, Academic Press, 2016, pp. 21-31, https://doi.org/10.1016/B978-0-12-800873-7.00002-6.

2. Heidari, M.D., Gandasasmita, S., Li, E., Pelletier, N. Proposing a framework for sustainable feed formulation for laying hens: A systematic review of recent developments and future directions, Journal of Cleaner Production, Vol. 288, 2021, https://doi.org/10.1016/j.jclepro.2020.125585 .

3. L.M. Wang, I.B. Mandell, B.M. Bohrer "Effects of feeding essential oils and benzoic acid to replace antibiotics on finishing beef cattle growth, carcass characteristics, and sensory attributes, Applied" Animal Science, Vol.36 (2), 2020, pp. 145-156, https://doi.org/10.15232/aas.2019-01908

4. Khalid A. Abdoun, Gamaleldin M. Suliman, Ahmed A. Alsagan, Osman A. Altahir, Mohammed Y. Alsaiady, Elfadil E. Babiker, Mohammed A. Al-Badwi, Faisal A. Alshamiry, Ahmed A. Al-Haidary "Replacing alfalfa-based? total mixed ration with

Moringa leaves for improving carcass and meat quality characteristics in lambs", Heliyon, Vol. 10 (17), 2024, e36863, https://doi.org/10.1016/j.heliyon.2024.e36863.

5. Кіктєв, М Постановка та вирішення задачі оптимізації годівлі тварин. Технологічний аудит та резерви виробництва. 2013. - №6/2 (14). - С.8-11.

6. Khort, D., Kutyrev, A., Smirnov, I., Osypenko, V., Kiktev, N. Computer vision system for recognizing the coordinates location and ripeness of strawberries. *Communications in Computer and Information Science*, vol. 1158, 2020, 3rd International Conference on Data Stream and Mining and Processing, DSMP 2020, Lviv.- pp. 334 – 343,

7. Bian, P., Li, W., Jin, Y. *et al.* Ensemble feature learning for material recognition with convolutional neural networks. *J Image Video Proc.* **2018**, 64 (2018). https://doi.org/10.1186/s13640-018-0300-z

8. Xu, H., Han, Z., Feng, S. *et al.* Foreign object debris material recognition based on convolutional neural networks. *J Image Video Proc.* **2018**, 21 (2018). https://doi.org/10.1186/s13640-018-0261-2

References

1. Victor Suresh, A. "Feed formulation software", Editor(s): Sergio F. Nates, Aquafeed Formulation, *Academic Press*, 2016, pp. 21-31, https://doi.org/10.1016/B978-0-12-800873-7.00002-6.

2. Heidari, M.D., Gandasasmita, S., Li, E., Pelletier, N. Proposing a framework for sustainable feed formulation for laying hens: A systematic review of recent developments and future directions, *Journal of Cleaner Production*, Vol. 288, 2021, https://doi.org/10.1016/j.jclepro.2020.125585 .

3. Wang, L.M., Mandell, I.B., Bohrer, B.M. "Effects of feeding essential oils and benzoic acid to replace antibiotics on finishing beef cattle growth, carcass characteristics, and sensory attributes, Applied" *Animal Science*, Vol.36 (2), 2020, pp. 145-156, https://doi.org/10.15232/aas.2019-01908

4. Khalid A. Abdoun, Gamaleldin M. Suliman, Ahmed A. Alsagan, Osman A. Altahir, Mohammed Y. Alsaiady, Elfadil E. Babiker, Mohammed A. Al-Badwi, Faisal A. Alshamiry, Ahmed A. Al-Haidary "Replacing alfalfa-based? total mixed ration with Moringa leaves for improving carcass and meat quality characteristics in lambs", *Heliyon*, Vol. 10 (17), 2024, e36863, https://doi.org/10.1016/j.heliyon.2024.e36863.

5. Kiktev, M. Formulation and solution of the problem of optimizing animal feeding [Postanovka ta vyrishennya zadachi optymizatsiyi hodivli tvaryn]. *Technological audit and production reserves*, 2013. - No. 6/2 (14). - pp.8-11. (In Ukranian).

6. Khort, D., Kutyrev, A., Smirnov, I., Osypenko, V., Kiktev, N. Computer vision system for recognizing the coordinates location and ripeness of strawberries. *Communications in Computer and Information Science*, vol. 1158, 2020, 3rd International Conference on Data Stream and Mining and Processing, DSMP 2020, Lviv.- pp. 334 – 343,

7. Bian, P., Li, W., Jin, Y. *et al.* Ensemble feature learning for material recognition with convolutional neural networks. *J Image Video Proc.* **2018**, 64 (2018). https://doi.org/10.1186/s13640-018-0300-z

8. Xu, H., Han, Z., Feng, S. *et al.* Foreign object debris material recognition based on convolutional neural networks. *J Image Video Proc.* **2018**, 21 (2018). https://doi.org/10.1186/s13640-018-0261-2

VISUAL ASSESSMENT OF RAW MATERIALS IN PREMIX PRODUCTION USING OpenCV

N. Kiktev, M. Pravilov, O. Opryshko, O. Romashchuk, D. Lavinskiy

Abstract. *The quality control of premix components is a crucial stage of their production. It directly influences a number of final product characteristics, including homogeneity, granulometric properties, and the effectiveness of the premix ingredients. This study highlights the relevance of using computer vision based on the OpenCV library for image processing of salt samples, as well as the application of deep learning with TensorFlow for automated quality classification. The results indicate whether the proposed tools are suitable for an efficient quality control process, enabling a reduction of human factor influence and improving the accuracy of defect detection.*

Key words: ***computer vision, salt quality assessment, premix production, machine learning, OpenCV, TensorFlow***

Додаток А.

dir.py | script.py | test.py | preproccess.py

```python
import cv2
import numpy as np
import os

# Функція для обробки одного зображення
def preprocess_image(image_path):  1 usage
    image = cv2.imread(image_path)
    if image is None:
        print(f"Не вдалося завантажити зображення: {image_path}")
        return None

    # Нормалізація яскравості
    image = cv2.normalize(image, dst: None, alpha: 0, beta: 255, cv2.NORM_MINMAX)

    # Корекція контрасту
    lab = cv2.cvtColor(image, cv2.COLOR_BGR2LAB)
    l, a, b = cv2.split(lab)
    l = cv2.equalizeHist(l)
    lab = cv2.merge((l, a, b))
    image = cv2.cvtColor(lab, cv2.COLOR_LAB2BGR)

    # Перетворення у градації сірого
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

    # Гаусове згладжування
    blurred = cv2.GaussianBlur(gray, ksize: (5, 5), sigmaX: 0)

    # Порогова обробка для виділення комків та домішок
    _, thresh = cv2.threshold(blurred, thresh: 150, maxval: 255, cv2.THRESH_BINARY_INV)

    return thresh
```

```python
# Функція для обробки всіх зображень у директорії
def preprocess_folder(input_folder, output_folder):  1 usage
    os.makedirs(output_folder, exist_ok=True)

    for filename in os.listdir(input_folder):
        if filename.lower().endswith(('.jpg', '.png', '.jpeg')):
            file_path = os.path.join(input_folder, filename)
            processed_image = preprocess_image(file_path)

            if processed_image is not None:
                output_path = os.path.join(output_folder, filename)
                cv2.imwrite(output_path, processed_image)
                print(f"Оброблено: {filename}")

    print(f"✅ Обробка завершена: {input_folder} -> {output_folder}")

# Обробка зображень у папці 'obr'
input_folder = "/Users/nickpravilov/PyCharmMiscProject/data/obr"
output_folder = "/Users/nickpravilov/PyCharmMiscProject/data/obr_processed"

preprocess_folder(input_folder, output_folder)
```

Рис.А1. Програмний модуль для обробки зображень за допомогою OpenCV

dir.py | script.py | test.py | preproccess.py

```python
import os
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Input, Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping

# 📌 Шляхи до даних
train_dir = "data/train_processed"
val_dir = "data/val_processed"
test_dir = "data/test"

# 📌 Аугментація для навчальних даних
train_datagen = ImageDataGenerator(
    rescale=1.0 / 255.0,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True
)

val_datagen = ImageDataGenerator(rescale=1.0 / 255.0)

train_dataset = train_datagen.flow_from_directory(
    train_dir, target_size=(128, 128), batch_size=32, class_mode='binary'
)
val_dataset = val_datagen.flow_from_directory(
    val_dir, target_size=(128, 128), batch_size=32, class_mode='binary'
)

# 📌 Балансування класів
class_weight = {0: 1.0, 1: 1.5}
```

```python
# 📌 Побудова моделі (Виправлено `input_shape`)
model = Sequential([
    Input(shape=(128, 128, 3)),  # Оголошуємо вхідні дані
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),  # Запобігає перенавчанню
    Dense(1, activation='sigmoid')
])

# 📌 Компіляція моделі
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# 📌 Callback для ранньої зупинки
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

# 📌 Запуск навчання (Виправлено `workers`)
history = model.fit(
    train_dataset,
    epochs=5,  # Оптимально для уникнення перенавчання
    validation_data=val_dataset,
    class_weight=class_weight,
    callbacks=[early_stopping]
)

# 📌 Збереження моделі
model.save("salt_quality_model.keras")


# 📌 Функція для побудови графіків
def plot_training_history(history):  1 usage
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
```

```python
# 📌 Функція для побудови графіків
def plot_training_history(history):  1 usage
    acc = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']

    epochs = range(1, len(acc) + 1)

    plt.figure(figsize=(12, 5))

    # Графік точності
    plt.subplot( *args: 1, 2, 1)
    plt.plot( *args: epochs, acc, label='Train Accuracy')
    plt.plot( *args: epochs, val_acc, label='Validation Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.title('Training and Validation Accuracy')

    # Графік втрат
    plt.subplot( *args: 1, 2, 2)
    plt.plot( *args: epochs, loss, label='Train Loss')
    plt.plot( *args: epochs, val_loss, label='Validation Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.title('Training and Validation Loss')

    plt.show()


# 📌 Побудова графіків після навчання
plot_training_history(history)
```

dir.py script.py test.py preproccess.py

```
# 📌 Завантажуємо модель для тестування
model = load_model("salt_quality_model.keras")


# 📌 Функція для передбачення якості солі на тестових зображеннях
def predict_salt_quality(image_path, model):  1 usage
    image = load_img(image_path, target_size=(128, 128))  # Завантаження зображення
    image = img_to_array(image) / 255.0  # Нормалізація
    image = np.expand_dims(image, axis=0)  # Додаємо розмірність для моделі

    prediction = model.predict(image)[0][0]  # Отримуємо ймовірність
    label = "Якісна сіль" if prediction > 0.5 else "Неякісна сіль"

    return label, prediction


# 📌 Функція для перевірки всіх зображень у `test/`
def evaluate_test_folder(test_folder, model):  1 usage
    if not os.path.exists(test_folder):
        print(f"❌ Папка {test_folder} не знайдена!")
        return

    print("\n📌 Результати перевірки на тестових зображеннях:")
    for filename in os.listdir(test_folder):
        if filename.lower().endswith(('.jpg', '.png', '.jpeg')):
            image_path = os.path.join(test_folder, filename)
            label, prediction = predict_salt_quality(image_path, model)
            print(f"{filename}: {label} (Ймовірність: {prediction:.4f})")


# 📌 Виконуємо перевірку на тестових зображеннях
evaluate_test_folder(test_dir, model)

print("✅ Навчання завершено! 🚀")
```

Рис.А2. Навчання нейронної мережі в TensorFlow

Додаток Б.

1. **Визначення шляхів до директорій**:

- train_dir = "data/train_processed"
- val_dir = "data/val_processed"
- test_dir = "data/test"

2. **Аугментація навчальних даних** виконується за допомогою ImageDataGenerator, де застосовуються такі перетворення:

- **rescale=1./255** – нормалізація піксельних значень до діапазону [0, 1].
- **rotation_range=20** – випадкове обертання зображень на кут до 20 градусів.
- **width_shift_range=0.2** та **height_shift_range=0.2** – зсув по ширині та висоті до 20%.
- **horizontal_flip=True** – випадкове відображення зображень по горизонталі.

3. **Формування датасетів**:

- train_dataset і val_dataset створюються за допомогою flow_from_directory(), що автоматично розпізнає класи з підкаталогів high_quality і low_quality.

4. **Балансування класів**:

- Задається словник class_weight = {0: 1.0, 1: 1.5}, щоб компенсувати можливу нерівновагу між класами (наприклад, якщо зразків якісної солі більше, ніж неякісної)

3. Побудова та навчання нейронної мережі

1. **Архітектура моделі** побудована за допомогою Sequential API:

- **Input(shape=(128, 128, 3))** – визначає вхідний розмір для зображень.
- **Conv2D(32, (3, 3), activation='relu')** – перший згортковий шар для виділення ознак.
- **MaxPooling2D(2, 2)** – зменшує розміри зображень для зниження обчислювальних витрат.
- Процес повторюється з **64 фільтрами** у другому згортковому шарі.
- **Flatten()** – перетворює двовимірні дані у одновимірний вектор.
- **Dense(128, activation='relu')** – повнозв'язний шар для класифікаційних ознак.
- **Dropout(0.3)** – вимикає випадкові нейрони для запобігання перенавчанню.
- **Dense(1, activation='sigmoid')** – вихідний шар для бінарної класифікації.

2. **Компіляція моделі**:

- Оптимізатор **adam**, функція втрат **binary_crossentropy**, метрика **accuracy**.

3. **Навчання моделі**:

- Використовується функція **fit()** з параметром epochs=5 для запобігання перенавчанню.
- **EarlyStopping** автоматично зупиняє навчання при виявленні погіршення на валідаційних даних.

4. **Збереження моделі** у форматі .keras для подальшого використання.

4. Тестування моделі та передбачення якості солі

1. **Завантаження збереженої моделі** для тестування:

model = load_model("salt_quality_model.keras")

2. **Функція predict_salt_quality()** виконує передбачення якості солі:

- Завантажує зображення та змінює його розмір до 128x128 пікселів.
- Нормалізує значення пікселів.

- Отримує передбачення від моделі:
  - **> 0.5** – "Якісна сіль"
  - **≤ 0.5** – "Неякісна сіль"

3. **Функція evaluate_test_folder()** автоматично перевіряє всі зображення у директорії test, повертаючи результат класифікації для кожного файлу.

4. **Виведення результатів**:

- Для кожного зображення виводиться назва файлу, результат класифікації та ймовірність.

5. Візуалізація результатів навчання

1. **Побудова графіків** точності та втрат для аналізу ефективності навчання:

- **Графік точності** показує зміни точності на навчальних та валідаційних даних.
- **Графік втрат** демонструє зміну функції втрат, що допомагає виявити перенавчання.

Виклик функції побудови графіків:

plot_training_history(history)