Жасанды интеллектcі бар жоғары жүктелген ақпараттық жүйелер мамандығының, "Ақпараттық жүйелер" кафедрасының студенті Құдайбергенов Жалғасбай Асқарұлы

**"Энергетикалық функция моделін қолдану арқылы нейрондық желіні оқыту әдістері"**

**Аннотация**

Бұл мақалада TensorFlow кітапханасын пайдаланып, энергетикалық функция моделін (EBM) MNIST деректер жиынтығы бойынша жүзеге асыру көрсетілген. Модельдің негізгі компоненттері — энергетикалық функция моделі, буфер класы және негізгі EBM моделі.Энергетикалық функция моделі конволюциялық және тығыз қабаттарды қолданып, суреттердің энергетикалық бағасын есептейді. Буфер класы Ланжевен динамикасы арқылы жасанды суреттерді жаңартып, сақтайды. EBM моделі контрастты дивергенция (CD) әдісін қолданып, шынайы және жасанды суреттер арасындағы айырмашылықты минимизациялау арқылы оқытылады.Мақалада модельді оқыту және тестілеу қадамдары сипатталады. Шынайы суреттер кездейсоқ шу арқылы түрлендіріледі, ал жасанды суреттер буферден алынады. Модельді оптимизациялау үшін Adam алгоритмі қолданылады. Негізгі шығындар ретінде контрастты дивергенция шығыны және регуляризация шығыны есептеледі.Деректерді алдын ала өңдеу кезеңінде MNIST суреттері -1 мен 1 аралығына масштабталып, 32x32 өлшеміне кеңейтіледі. Оқыту нәтижесінде модель шынайы және жасанды суреттерді ажыратуды үйреніп, энергетикалық функцияны тиімді бағалауды жүзеге асырады.

В данной статье представлена реализация модели энергетической функции (EBM) на наборе данных MNIST с использованием библиотеки TensorFlow. Основные компоненты модели — модель энергетической функции, класс буфера и основная модель EBM.Модель энергетической функции использует сверточные и полносвязные слои для оценки энергетического значения изображений. Класс буфера обновляет и сохраняет искусственные изображения с помощью Ланжевенской динамики. Модель EBM обучается путем минимизации разницы между реальными и искусственными изображениями с использованием метода контрастной дивергенции (CD).В статье описываются шаги обучения и тестирования модели. Реальные изображения модифицируются путем добавления случайного шума, а искусственные изображения выбираются из буфера. Для оптимизации используется алгоритм Adam. Основными потерями являются потеря контрастной дивергенции и потеря регуляризации.На этапе предварительной обработки изображения MNIST масштабируются до диапазона от -1 до 1 и расширяются до размера 32x32. В результате обучения модель учится различать реальные и искусственные изображения и эффективно оценивать энергетическую функцию.

This article presents the implementation of an Energy-Based Model (EBM) using the TensorFlow library on the MNIST dataset. The primary components of the model are the energy function model, the buffer class, and the main EBM model.The energy function model employs convolutional and dense layers to estimate the energy values of images. The buffer class updates and stores synthetic images using Langevin dynamics. The EBM model is trained by minimizing the difference between real and synthetic images using the contrastive divergence (CD) method.The article details the training and testing steps of the model. Real images are perturbed with random noise, while synthetic images are sampled from the buffer. The model is optimized using the Adam algorithm. The primary losses considered are the contrastive divergence loss and the regularization loss.During the preprocessing phase, MNIST images are scaled to the range [-1, 1] and resized to 32x32. As a result of training, the model learns to distinguish between real and synthetic images and effectively estimate the energy function.

**Кілт сөздер**

Кілт сөздер: Энергетикалық функция моделі (EBM),TensorFlow,MNIST деректер жиынтығы,Ланжевен динамикасы,Контрастты дивергенция (CD),Жасанды суреттер,Сверточты нейрондық желілер (CNN),Оқыту және тестілеу,Деректерді алдын ала өңдеу,Регуляризация шығыны,Шынайы суреттер,Шығындар функциясы,Оптимизация,Суреттердің энергетикалық бағасы,Жасанды нейрондық желілер

**Кіріспе**

Энергетикалық функция модельдері (Energy-Based Models, EBM) ықтималдық модельдеу мен генеративті нейрондық желілер саласындағы маңызды және қуатты әдістердің бірі. Олар деректерді ықтималдық функциялары арқылы емес, энергия функциялары арқылы сипаттайды. EBM-дер жоғары ықтималдықтағы деректерге төмен энергия мәнін тағайындайды, ал ықтималдығы төмен деректерге жоғары энергия мәнін тағайындайды. Бұл әдіс арқылы күрделі деректерді, мысалы, суреттер, мәтіндер немесе уақыттық қатарлар сияқты құрылымды ақпаратты модельдеуге мүмкіндік береді.EBM-нің басты артықшылықтарының бірі – олардың икемділігі мен әмбебаптығы. Олар ықтималдық таралулардың нақты түріне тәуелсіз және көптеген түрлі деректер типтерін модельдей алады. Сонымен қатар, EBM-дер дискриминативті (ажыратушы) және генеративті (генерациялаушы) тапсырмаларда тиімді қолданылады.

Бұл мақалада TensorFlow кітапханасын пайдаланып, MNIST деректер жиынтығы бойынша энергетикалық функция моделін жүзеге асыру көрсетіледі. MNIST – қолмен жазылған сандардың суреттерінен тұратын классикалық деректер жиынтығы, ол машиналық оқыту мен нейрондық желілерді зерттеуде жиі қолданылады. Модельді оқыту процесінде контрастты дивергенция (Contrastive Divergence, CD) әдісі қолданылады. CD әдісі арқылы модель шынайы суреттер мен жасанды түрде генерацияланған суреттер арасындағы айырмашылықты минимизациялауды үйренеді.

Жасанды суреттерді генерациялау үшін Ланжевен динамикасы қолданылады. Бұл әдіс кездейсоқ шумен және градиенттермен суреттерді итеративті түрде жаңартады. Ланжевен динамикасы EBM-ге жаңа үлгілерді үздіксіз жақсартып отыруға мүмкіндік береді.Мақалада энергетикалық функция моделі үш негізгі компоненттен тұрады:Энергетикалық функция моделі – конволюциялық және тығыз қабаттарды қолданып, суреттердің энергетикалық бағасын есептейді.Буфер класы – жасанды түрде генерацияланған суреттерді сақтайды және оларды Ланжевен динамикасы арқылы жаңартады.EBM моделі – модельді оқыту процесін басқарады және контрастты дивергенция шығынын минимизациялау арқылы шынайы және жасанды суреттер арасындағы айырмашылықты азайтады.Деректерді алдын ала өңдеу кезеңінде MNIST суреттері -1 мен 1 аралығына масштабталады және 32x32 өлшеміне кеңейтіледі. Модельді оптимизациялау үшін Adam алгоритмі қолданылады, бұл градиенттерді жылдам және тұрақты түрде есептеуге мүмкіндік береді.

Оқу процесінің соңында EBM шынайы және жасанды суреттерді ажыратуды үйреніп, энергетикалық функцияны тиімді бағалауды жүзеге асырады. Бұл әдіс генеративті модельдеудің көптеген салаларында, соның ішінде бейнелерді генерациялау, аномалияларды анықтау және ықтималдық бағалауда қолданылады.

**Негізгі бөлім**

### **1. Энергетикалық функция моделін құру**

Бұл бөлімде энергетикалық функция моделін құру үшін нейрондық желі анықталады. Модель кіріс ретінде сұр түсті суреттерді қабылдайды және әр қабатта конволюция операциясы орындалады. Модельдің әр қабаты мен ядросы суреттердің ерекшеліктерін шығарып, соңғы қабатта энергетикалық мән шығарады. Бұл мән модельдің нақты немесе жасанды суреттің қаншалықты дұрыс екендігін бағалайды.

def build\_energy\_model():

 model = tf.keras.Sequential([

 layers.Input(shape=(32, 32, 1)), # 32x32 өлшемді және 1 арналы (сұр түсті) суреттер үшін кіріс қабаты

 layers.Conv2D(16, kernel\_size=5, strides=2, padding="same", activation="swish"), # 16 ядролы бірінші конволюциялық қабат

 layers.Conv2D(32, kernel\_size=3, strides=2, padding="same", activation="swish"), # 32 ядролы екінші конволюциялық қабат

 layers.Conv2D(64, kernel\_size=3, strides=2, padding="same", activation="swish"), # 64 ядролы үшінші конволюциялық қабат

 layers.Conv2D(64, kernel\_size=3, strides=2, padding="same", activation="swish"), # Қайталанған 64 ядролы конволюциялық қабат

 layers.Flatten(), # Конволюциялық қабаттың нәтижесін тегістеу

 layers.Dense(64, activation="swish"), # Тығыз қабат, 64 нейроннан тұрады

 layers.Dense(1) # Энергетикалық бағалау үшін бір нейрондық шығыс қабаты

 ])

 return model

Бұл кодта біз *кернеулі нейрондық желі* құрып, суреттерді өңдеуге және олардың энергетикалық бағасын есептеуге дайынбыз.

### **2. Буфер класын анықтау**

Буфер класы жасанды суреттерді сақтау және оларды генерациялау үшін өте маңызды рөл атқарады. Бұл класс нақты суреттердің үлгілерін жинақтайды және жаңа суреттерді генерациялау үшін оларды түрлі әдістермен өңдейді. Сонымен қатар, буфер арқылы суреттердің жаңару процесі бақыланып, Ланжевен динамикасы сияқты әдістермен олар жақсартылып отырады. Осылайша, жасанды суреттердің сапасын арттыруға және деректердің әртүрлілігін көбейтуге мүмкіндік береді.

class Buffer:

 def \_\_init\_\_(self, model):

 self.model = model

 # Алғашқы буферге 128 кездейсоқ сурет жүктеледі

 self.examples = [tf.random.uniform((1, 32, 32, 1)) \* 2 - 1 for \_ in range(128)] # Суреттер [-1, 1] диапазонында болады

 # Ланжевен динамикасы арқылы жаңа суреттерді генерациялау функциясы

 def sample\_new\_exms(self, steps, step\_size, noise):

 """

 Буфердегі суреттерді жаңарту және жаңа суреттерді генерациялау.

 - steps: Ланжевен динамикасының қадам саны

 - step\_size: Жаңарту қадамының мөлшері

 - noise: Шу деңгейі

 """

 # Буфердегі суреттердің 5%-ін кездейсоқ жаңартып отыру

 n\_new = np.random.binomial(128, 0.05) # Кездейсоқ 5% жаңа сурет

 rand\_imgs = tf.random.uniform((n\_new, 32, 32, 1)) \* 2 - 1 # Кездейсоқ суреттер генерациясы

 old\_imgs = tf.concat(random.choices(self.examples, k=128 - n\_new), axis=0) # Ескі суреттерді таңдау

 inp\_imgs = tf.concat([rand\_imgs, old\_imgs], axis=0) # Жаңа және ескі суреттерді біріктіру

 # Ланжевен динамикасы арқылы суреттерді жақсарту

 for \_ in range(steps):

 inp\_imgs += tf.random.normal(inp\_imgs.shape, mean=0, stddev=noise) # Шу қосу

 inp\_imgs = tf.clip\_by\_value(inp\_imgs, -1.0, 1.0) # Суреттердің мәнін [-1, 1] аралығында шектеу

 with tf.GradientTape() as tape:

 tape.watch(inp\_imgs)

 out\_score = -self.model(inp\_imgs) # Энергетикалық мәндерді есептеу

 grads = tape.gradient(out\_score, inp\_imgs) # Градиенттерді есептеу

 grads = tf.clip\_by\_value(grads, -0.03, 0.03) # Градиенттерді шектеу

 inp\_imgs += -step\_size \* grads # Суреттерді жаңарту

 inp\_imgs = tf.clip\_by\_value(inp\_imgs, -1.0, 1.0) # Мәндерді шектеу

 # Буферді жаңартып, максималды өлшемін 8192 суреттен асырмайды

 self.examples = tf.split(inp\_imgs, 128, axis=0) + self.examples # Жаңартылған суреттерді буферге қосу

 self.examples = self.examples[:8192] # Буфердің максималды өлшемін шектеу

 return inp\_imgs # Жаңартылған суреттерді қайтару

###

### *Буфер классы мен Ланжевен динамикасының маңызы*

Ланжевен динамикасы: Бұл әдіс суреттерді жақсартуда өте пайдалы. Ланжевен динамикасы шулы градиенттермен жұмыс істейді және бұл әдіс суреттердің жаңа ерекшеліктерін жасауға мүмкіндік береді.

Буфердің жаңартылуы: Жасанды суреттерді жаңарту және оларды әртүрлілікпен қамтамасыз ету мақсатында буферді жаңарту маңызды. Бұл жасанды деректердің әртүрлілігін сақтауға және оларды модельді оқыту үшін жақсырақ пайдалануға мүмкіндік береді.

Буфердің бұл жұмысы жасанды деректердің санын көбейтуге және олардың сапасын жақсартуға, сондай-ақ модельдің дәлдігін арттыруға көмектеседі.

### **3. EBM моделін анықтау**

Энергетикалық Базельдік Модель (EBM) объектісі нақты және жасанды суреттерді қабылдап, олардың энергетикалық бағасын есептейді. Бұл әдіс контрастты дивергенция (contrastive divergence) тәсілін пайдаланып, модельдің қателіктерін минимизациялайды. Mодель оқыту барысында оң және теріс үлгілер арасындағы айырмашылықты түсінеді.

class EBM(Model):

 def \_\_init\_\_(self, model, buffer, alpha=0.1, learning\_rate=0.0001):

 super(EBM, self).\_\_init\_\_()

 self.model = model

 self.buffer = buffer

 self.alpha = alpha

 # Модельді оптимизациялау үшін Adam оптимизаторы қолданылады

 self.optimizer = optimizers.Adam(learning\_rate=learning\_rate)

 # Модельдің оқыту қадамы

 def train\_step(self, real\_imgs):

 # Нақты суреттерге аздаған шу қосылады

 real\_imgs += tf.random.normal(tf.shape(real\_imgs), mean=0, stddev=0.005)

 real\_imgs = tf.clip\_by\_value(real\_imgs, -1.0, 1.0)

 # Буферден жасанды суреттер алынады

 fake\_imgs = self.buffer.sample\_new\_exms(steps=60, step\_size=10, noise=0.005)

 inp\_imgs = tf.concat([real\_imgs, fake\_imgs], axis=0)

 with tf.GradientTape() as tape:

 real\_out, fake\_out = tf.split(self.model(inp\_imgs), 2, axis=0)

 # Контрастты дивергенция және регуляризация шығынын есептеу

 cdiv\_loss = tf.reduce\_mean(fake\_out) - tf.reduce\_mean(real\_out)

 reg\_loss = self.alpha \* tf.reduce\_mean(real\_out \*\* 2 + fake\_out \*\* 2)

 loss = reg\_loss + cdiv\_loss

 # Градиенттер арқылы модель параметрлерін жаңарту

 grads = tape.gradient(loss, self.model.trainable\_variables)

 self.optimizer.apply\_gradients(zip(grads, self.model.trainable\_variables))

 return {"loss": loss}

EBM моделі *нақты және жасанды суреттер арасындағы айырмашылықты нақты бағалап, осы айырмашылықтарды минимизациялайды*, бұл оқыту сапасын арттырады.

### **4. Деректерді жүктеу және алдын ала өңдеу**

Бұл бөлімде MNIST деректер жинағы жүктеледі және өңделеді. MNIST — 0-ден 9-ға дейінгі цифрлардың сұр түсті суреттерінен тұратын кеңінен қолданылатын деректер жиынтығы. Әрбір сурет 28x28 өлшемінде болады және пиксель мәндері [0, 255] аралығында сақталады.

Модель үшін деректерді тиімді пайдалану үшін келесі алдын ала өңдеу қадамдары орындалады:

1. Масштабтау: Пиксель мәндерін [-1, 1] диапазонына келтіру.
2. Өлшемді өзгерту: 28x28 өлшемін 32x32 өлшеміне дейін ұлғайту үшін жиекке -1 мәнін қосу.
3. Өлшем қосу: Тізбектелген нейрондық желілер үшін деректерді 4 өлшемді массивке айналдыру (мысалы, (сурет саны, 32, 32, 1)).

(x\_train, \_), (x\_test, \_) = tf.keras.datasets.mnist.load\_data()

def preprocess(imgs):

 imgs = (imgs.astype("float32") - 127.5) / 127.5 # Пиксель мәндерін [-1, 1] диапазонына масштабтау

 imgs = np.pad(imgs, ((0, 0), (2, 2), (2, 2)), constant\_values=-1.0) # Суреттерді 32x32 өлшеміне кеңейту

 imgs = np.expand\_dims(imgs, -1) # Суреттерді 4 өлшемді массивке айналдыру

 return imgs

x\_train = preprocess(x\_train)

x\_test = preprocess(x\_test)

x\_train = tf.data.Dataset.from\_tensor\_slices(x\_train).batch(128)

x\_test = tf.data.Dataset.from\_tensor\_slices(x\_test).batch(128)

Бұл өңдеу қадамдары модельдің MNIST деректер жиынтығымен дұрыс жұмыс істеуін қамтамасыз етеді. Суреттерді масштабтау және өлшемін өзгерту арқылы модельге енгізілетін деректер стандартталады, ал батчтарға бөлу оқыту тиімділігін арттырады.

### **5. Модельді құру және оқыту**

Бұл бөлімде нейрондық желі моделі құрылады және 60 эпох бойынша оқытылады. Оқыту процесінде модель кіріс деректерін өңдеп, болжамдар жасайды және шығын (loss) мәнін азайтуға тырысады. Әр эпох сайын шығындар есептеліп, модельдің нәтижелері жақсарады. Сондай-ақ, тексеру деректеріне (validation data) негізделген нәтижелерді бақылау арқылы модельдің артық үйренуін (overfitting) болдырмауға болады.

Модельді құру

# Модельді құру

def create\_model():

 model = models.Sequential([

 layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(32, 32, 1)),

 layers.MaxPooling2D((2, 2)),

 layers.Conv2D(64, (3, 3), activation='relu'),

 layers.MaxPooling2D((2, 2)),

 layers.Flatten(),

 layers.Dense(128, activation='relu'),

 layers.Dense(10, activation='softmax') # 10 сынып үшін softmax шығысы

 ])

 return model

1. 2D Сверткіш қабаттар (Conv2D): Суреттерден ерекшеліктерді (features) анықтайды.
2. Pooling қабаттар (MaxPooling2D): Суреттердің өлшемін азайтып, есептеу жүктемесін төмендетеді.
3. Тығыз қабаттар (Dense): Сверткіш қабаттар анықтаған ерекшеліктерді талдап, нәтижесін шығарады.
4. Softmax шығысы: 10 сыныптың біреуін таңдау үшін пайдаланылады (0-ден 9-ға дейінгі цифрлар).

*Буфер мен EBM инициализациясы*

buffer = Buffer(model)

ebm = EBM(model=model, buffer=buffer)

Buffer: Модельдің аралық нәтижелерін сақтау үшін қажет.

EBM (Energy-Based Model): Модельді энергия негізіндегі әдіспен оқыту үшін қолданылады.

 Модельді жинақтау

ebm.compile(optimizer=tf.keras.optimizers.Adam(learning\_rate=0.001),

 loss='sparse\_categorical\_crossentropy',

 metrics=['accuracy'],

 run\_eagerly=True)

Adam оптимизаторы: Градиенттерді тиімді есептеу үшін қолданылады.

Шығын функциясы: *sparse\_categorical\_crossentropy* — санаттарды сандық түрде көрсеткен кезде қолданылады.

run\_eagerly=True: Модельді eager режимінде орындауға мүмкіндік береді (дебаг үшін пайдалы).

*Оқыту процесі*

history = ebm.fit(x\_train, epochs=60, validation\_data=x\_test)

60 эпох: Модель 60 рет толық деректер жиынтығы бойынша оқытылады.

Тексеру деректері: Оқыту кезінде модельдің тексеру деректеріндегі өнімділігі бақыланады**.**

### *Оқыту нәтижесін визуализациялау*

# Оқыту және тексеру дәлдігін визуализациялау

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.title('Training and Validation Accuracy')

plt.show()



Графикте оқыту процесіндегі шығындардың өзгерісі көрсетілген:

1. Loss (Жалпы шығын) – модельдің жалпы шығын функциясы. Әрбір эпохада төмендеп, оқыту барысында жақсарып жатқанын көрсетеді.
2. Reg (Регуляризация шығыны) – модельдің параметрлерін шамадан тыс өсуіне жол бермеу үшін енгізілген регуляризация шығыны. Бұл шығынның төмендеуі модельдің тұрақтылығын арттыруды білдіреді.
3. Cdiv (Контрастты дивергенция шығыны) – модельдің шынайы және жасанды деректер арасындағы айырмашылықты жақсарту үшін қолданылатын шығын. Бұл көрсеткіш те оқыту барысында азайып, модельдің тиімді оқытылып жатқанын көрсетеді.

###

###

###

###

###

###

###

### **Қорытынды**

Бұл мақалада MNIST деректер жинағын пайдалана отырып, нейрондық желі моделін құру және оны оқыту процесі толық қарастырылды. Жұмыс барысында компьютерлік көру міндеттерін шешуге бағытталған маңызды кезеңдер сипатталды, оның ішінде деректерді алдын ала өңдеу, модельді құру және оқыту процесі ерекше назарға алынды.Модельді сәтті оқыту үшін деректерді сапалы түрде алдын ала өңдеу маңызды рөл атқарады. MNIST деректер жиынтығындағы суреттер 28x28 пиксель өлшемінде болғандықтан, оларды нейрондық желінің қажеттіліктеріне сәйкес 32x32 өлшеміне кеңейту жүргізілді. Сондай-ақ пиксель мәндері [-1, 1] диапазонына масштабталды, бұл модельдің тиімділігін арттырып, оқыту процесін тұрақтандырды. Деректер tf.data.Dataset көмегімен батчтарға бөлініп, оқытуға ыңғайлы форматқа келтірілді.Жұмыста convolutive нейрондық желі (CNN) моделі құрылды. Бұл модель суреттердегі маңызды ерекшеліктерді тиімді түрде анықтау үшін бірнеше сверткіш (Conv2D) және pooling қабаттарын қолданды. Ақырғы softmax қабаты модельге 0-ден 9-ға дейінгі цифрларды классификациялауға мүмкіндік берді.Модельді оқыту үшін Adam оптимизаторы және sparse\_categorical\_crossentropy шығын функциясы қолданылды. Оқыту 60 эпох бойы жүргізіліп, әр кезеңде қателік көрсеткіштері мен дәлдік мәндері бақыланды. Оқыту барысында тексеру (validation) деректері қолданылып, модельдің артық үйрену (overfitting) мәселесі туындамауын қамтамасыз еттік.Оқыту процесінің нәтижелері дәлдік пен шығын графиктері арқылы талданды. Оқыту деректері бойынша жоғары дәлдікке қол жеткізілді, ал тексеру деректері арқылы модельдің жалпы қабілеті бағаланды. Бұл модель MNIST деректер жиынтығындағы цифрларды тану міндетін сәтті орындап, жақсы өнімділік көрсетті.Бұл жұмыс нәтижесінде алынған нейрондық желі моделін басқа компьютерлік көру міндеттерінде қолдануға болады. Деректерді алдын ала өңдеу мен модель құру әдістерін бейімдеп, күрделі деректер жиынтықтарымен жұмыс істеуге мүмкіндік бар. Сонымен қатар, бұл әдісті цифрларды танудан бөлек, қолжазбаларды тану, объектілерді классификациялау және басқа да салаларда қолдануға болады.Жалпы, мақалада сипатталған тәсілдер мен әдістер нейрондық желілерді тиімді қолдануға және оларды нақты міндеттерде пайдалануға жол ашады.

### **Қолданылған әдебиеттер**

1. LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). *Gradient-based learning applied to document recognition*. Proceedings of the IEEE, 86(11), 2278-2324. [MNIST деректер жинағын сипаттайтын классикалық мақала].
2. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. [Нейрондық желілердің архитектурасы мен оқыту әдістеріне терең шолу].
3. TensorFlow Documentation.<https://www.tensorflow.org>. [TensorFlow кітапханасының ресми құжаттамасы].
4. Keras Documentation.<https://keras.io>. [Keras API және оны нейрондық желілерді құру үшін қолдану туралы нұсқаулық].
5. Deng, L. (2012). *The MNIST Database of Handwritten Digit Images for Machine Learning Research*. IEEE Signal Processing Magazine, 29(6), 141-142. [MNIST деректер жиынтығын машиналық оқыту зерттеулерінде қолдану].