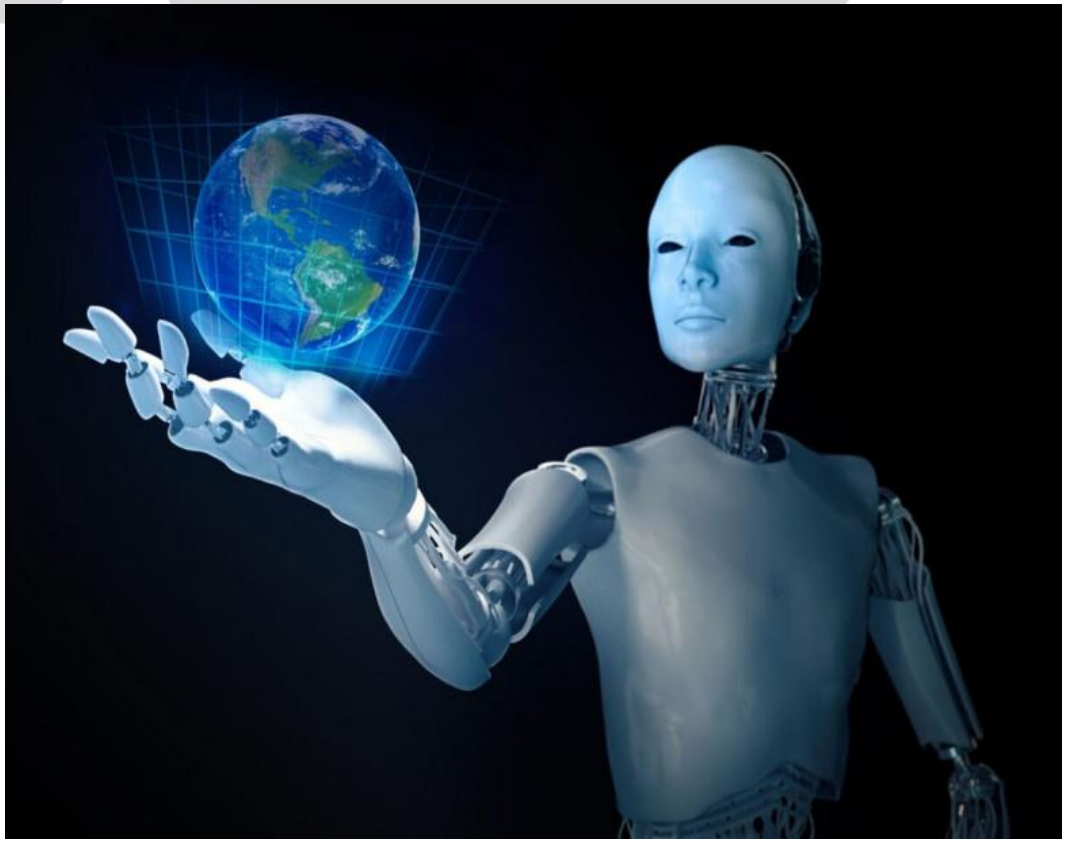
Министерство образования Красноярского края
Краевое государственное бюджетное профессиональное образовательное учреждение
«Красноярский колледж радиоэлектроники и информационных технологий»

Искусственный интеллект в образовательном процессе: за и против

Что такое искусственный интеллект (ИИ)?



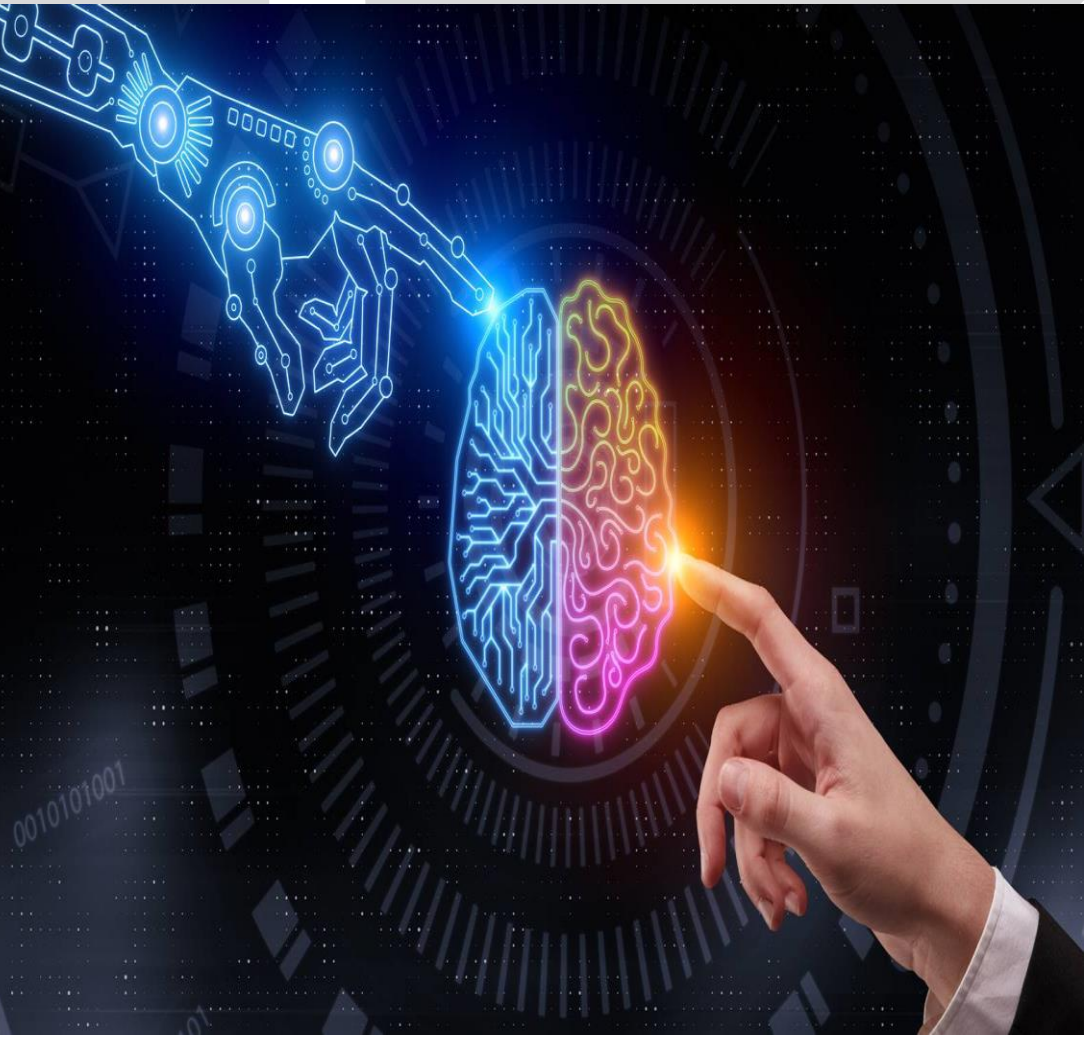
Искусственный интеллект (ИИ) - это область компьютерной науки, которая занимается разработкой систем и алгоритмов, способных выполнять задачи, традиционно требующие человеческого интеллекта, такие как распознавание образов, принятие решений, обучение и адаптация

Преимущества и недостатки ИИ в образовании

- Минусы:
- 1. Необходимость значительных инвестиций в инфраструктуру и разработку ИИ-систем.
- 2. Риски ошибок и предвзятости алгоритмов, которые могут приводить к несправедливым решениям.
- 3. Возможное снижение роли преподавателей и личного взаимодействия.
- 4. Этические вопросы, связанные с использованием данных учащихся и конфиденциальностью.
- 5. Сложность внедрения ИИ-технологий в традиционные образовательные системы.

- Плюсы:
- 1. Персонализация и адаптация обучения под каждого учащегося.
- 2. Повышение вовлеченности и мотивации учащихся за счет адаптивного контента.
- 3. Возможность предоставления непрерывной обратной связи и рекомендаций.
- 4. Высвобождение времени преподавателей за счет автоматизации рутинных задач.
- 5. Улучшение аналитики и принятия решений на основе данных об обучении.

Основные области применения ИИ в образовании:



- - Адаптивное обучение: ИИ-системы анализируют успеваемость и поведение учащихся, чтобы подстраивать содержание и методы под индивидуальные потребности.
- - Интеллектуальные обучающие системы: ИИ-программы, которые могут выступать в роли "виртуальных репетиторов", предоставляя персонализированную обратную связь и рекомендации.
- - Автоматизация рутинных задач: ИИ может упростить процессы оценки, обратной связи, администрирования и других повторяющихся задач преподавателей.
- - Анализ данных об обучении: ИИ-алгоритмы позволяют выявлять закономерности в данных об успеваемости, посещаемости и других аспектах, чтобы улучшать образовательный процесс.

Transformer и RNN (Recurrent Neural Network)

- Transformer - это архитектура, на которой основаны многие современные модели, включая GPT и BERT. Transformer использует механизмы внимания, что позволяет эффективно обрабатывать последовательности данных, такие как текст, и учитывать контекст слов.


```
model = tf.keras.Sequential()
# Добавим слой Embedding ожидая на входе словарь размера 1000, и
# на выходе вложение размерностью 64.
model.add(layers.Embedding(input_dim=1000, output_dim=64))

# Добавим слой LSTM с 128 внутренними узлами.
model.add(layers.LSTM(128))

# Добавим слой Dense с 10 узлами и активацией softmax.
model.add(layers.Dense(10))

model.summary()
```

- RNN (Recurrent Neural Network) - нейросеть хорошо справляется с задачами, связанными с предсказанием следующего элемента в последовательности, например, в чат-ботах или при обработке текстов

RNN не всегда справляются с долгосрочными зависимостями. Есть решение: перейти на Transformer. Обрабатывается вся последовательность одновременно, благодаря механизму внимания. Это значительно ускорило обучение и улучшило качество их результатов.

LSTM (Long Short-Term Memory) и Gated Recurrent Unit (GRU)

- GRU — это тип рекуррентной нейронной сети (RNN), который был разработан для решения проблемы затухающего градиента и улучшения обработки последовательных данных. GRU имеет менее сложную архитектуру по сравнению с LSTM, но при этом сохраняет многие их преимущества.

```python
from keras.models import Sequential
from keras.layers import GRU, Dense, Embedding, GlobalMaxPooling1D

model = Sequential()
model.add(Embedding(input_dim=vocab_size, output_dim=embedding_dim, input_length=max_l
model.add(GRU(units=128, return_sequences=True))
model.add(GlobalMaxPooling1D())
model.add(Dense(units=1, activation='sigmoid'))  # Для бинарной классификации

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

LSTM - это разновидность RNN, которая лучше справляется с долгосрочными зависимостями в данных. LSTM часто используется в задачах, связанных с временными рядами или обработкой последовательностей, таких как машинный перевод или анализ текста.

```python
from keras.models import Sequential
from keras.layers import LSTM, Dense, Dropout

model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(timesteps, features)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=1))  # Выходной слой для предсказания температуры

model.compile(optimizer='adam', loss='mean_squared_error')
```

- GRU проще настраивать и ИИ быстрее обучаются в некоторых случаях. Это позволило быстрее получать результаты в проектах с короткими последовательностями.

Autoencoder и PCA (Principal Component Analysis)

- PCA (Principal Component Analysis) — это метод уменьшения размерности, который используется для анализа и визуализации данных. Он не является нейросетью, но его концепции могут быть реализованы с помощью нейросетей, например, через автоэнкодеры.

Применение PCA:

python — Копировать код

```python
from sklearn.decomposition import PCA
import numpy as np

# Предположим, X — это ваша матрица изображений (число образцов x 4096)
pca = PCA(n_components=2)  # Снижение до 2 компонентов
X_reduced = pca.fit_transform(X)
```

Autoencoder: Эта нейросеть используется для кодирования данных в более компактное представление и последующего восстановления исходных данных. Autoencoder часто применяется для сжатия изображений или в задачах, связанных с уменьшением размерности.

Создание модели автоэнкодера:

python — Копировать код

```python
from keras.layers import Input, Dense
from keras.models import Model

input_img = Input(shape=(784,))  # Для MNIST, 28x28 = 784
encoded = Dense(64, activation='relu')(input_img)  # Кодировщик
decoded = Dense(784, activation='sigmoid')(encoded)  # Декодировщик

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
```

Применение PCA для уменьшения размерности данных с ограничениями линейности этого метода. Autoencoder смог захватить более сложные структуры в их данных, обеспечивая лучшую реконструкцию и представление.

Нейросети	Преимущества	Недостатки
Transformer	- Высокая производительность на длинных последовательностях - Параллелизация и быстрый тренинг - Эффективен для обработки текстов и изображений	- Требует много данных для обучения - Высокие вычислительные ресурсы
RNN	- Подходит для последовательных данных (временные ряды, текст) - Способен учитывать предыдущие состояния	- Проблема исчезающего градиента - Сложности с длинными зависимостями
LSTM	- Устойчив к проблеме исчезающего градиента - Эффективен для долгосрочных зависимостей	- Сложность модели - Долгое время обучения
GRU	- Меньше параметров, чем LSTM - Быстрее обучается и требует меньше памяти	- Может не так хорошо справляться с длинными последовательностями, как LSTM
Autoencoder	- Эффективен для снижения размерности - Может использоваться для устранения шума	- Требует большого количества данных для обучения - Может не всегда извлекать значимые признаки
PCA	- Простота и быстрота вычислений - Эффективен для визуализации данных	- Линейность метода (может не учитывать нелинейные зависимости) - Потеря информации при снижении размерности

Использование студентами и преподавателями описанных ИИ в обучении и работе

Опрашиваемые	Кол-во опрошенных	Использую	Не использую
Студенты 1 курса	20	64%	36%
Студенты 2 курса	17	88%	12%
Студенты 3 курса	11	97%	3%
Преподаватели	7	80%	20%